Hands-on Machine Learning for Cybersecurity

Safeguard your system by making your machines intelligent using Python ecosystem

By Soma Halder

Packt
www.packt.com

Table of Contents

Preface

Chapter 1, Basics of Machine Learning in Cyber Security

The key focus of the chapter is allow readers to get familiarized with machine learning, how is it practiced and its need in Cyber Security domain. This chapter builds on using machine learning instead of conventional rule based engines while allowing the readers to tackle challenges in the cyber security domain. Further, this chapter allow readers to get hands-on knowledge on python, MongoDB and various libraries for machine learning and concepts related to supervised and unsupervised learning.

Chapter 2, Time series Analysis and Ensemble Modeling

The first phase of the threat lifecycle deals with Reconnaissance where malwares/APTS passively engages the target by searching through public information, penetrating confidential corporate documents and so on. Time series Analysis helps detect packets exchanged at odd hours or identify spikes seen during holidays or odd hours in the corporate network.Given a sequence numbers for time series dataset we can restructure the data model to look like a supervised learning by using the values at previous point in time to predict the value at next point. Time
Similarly Ensembles methods are techniques that give boost in accuracy on predictions made by our model. Meta learning from the ensemble algorithm helps identify the resources which are getting exploited to continue the reconnaissance activity.

In this chapter we will implement two examples on ensemble modeling and time series forecasting each.

Chapter 3, Segregating Legitimate and Lousyc

In the threat kill chain the initial attack often starts with “URL Injection” in emails, attachments etcetera. Detecting bad urls in the initial stages of the attack help security professionals to combat them early on. Thus, Having learnt the base concepts of machine learning this chapter will focus on building practical skills and will discuss examples of machine learning Implementation on URLs, identifying the good, bad and the worst of URLs through an intelligent machine learning based python driven example application. Additionally, this chapter will provide hands-on knowledge to the concepts learnt in the first chapter as well as testing the accuracy of result generated by our program.

Chapter 4, Knocking Down Captchas

In the previous chapter, we learnt about defensive mechanisms like URL analysis and building intelligent programs that can detect anomalies in the web request/response. In this chapter, we will learn how attackers can effectively bypass defensive technologies using machine learning. This chapter will discuss bypassing Captcha mechanism through machine learning and will build

on hands-on knowledge of image processing, voice and text processing in order to bypass Captcha. At the end of this chapter, the readers will have hands-on knowledge on defeating protection mechanisms while learning how they can build effective countermeasures.

Chapter 5, Using Data Science to Catch Email Frauds and Spams

In the last three chapters, we have covered the basics of ML while putting it use in cyber security. This chapter will focus on real world scenarios and case studies. A steep rise in the complexity of cyber attacks has made it very difficult for the blue teams to develop a proper countermeasure to fraudulent emails and spam. In this chapter, we will try to build a program that will solve the problem of detecting email frauds and spam using supervised learning with the Naive Bayes algorithm. The program also enables learning the insights of Email, its headers and distinct information for fraud and spam classification.

Chapter 6, Efficient Network Anomaly detection using K Means

Network Anomalies could be pointers to malware's connectivity with the command and control, the malware's lateral movement and exfiltration from with the network. Thus accurately detecting network anomalies can help the network administrator to prevent a cyber attack. We will cover alert and exploration based network security monitoring in this chapter. Making effective use of K Means algorithm, we will try developing a solution which will have the ability to detect anomalies and alert any malicious activity going on the network. However, searching for anomalous network flows is a challenging task due to the wide statistical variety of network behavior. The chapter will also allow readers to gain knowledge on Python's SK-Learn and NumPy set of libraries and modules.

Chapter 7, Decision Tree and Context Based Malicious Event Detection

Making decisions are hard, whether it's life changing decisions or the decisions on types of events occurring in a system. This chapter develop reader's skill on defining better decision making using decision tree. Decision trees are predictive models designed to go from observation to conclusion. We know that each event or incident in computer system's security consists of number of independent series of actions. However, using only machine learning and processing each individual action on a standalone basis, we can not detect if it was malicious or not. Therefore, we will learn about preserving context of the previous events that can help us to take decision about the credibility of actions in particular event.

Chapter 8, Catching Impersonators and Hackers Red Handed

Each user has their different digital fingerprint that includes how we interact with some application, our typing speed, way we move our hand on touch screen or how we scroll on a web page etc. Machine learning gives us the ability to transparently produce a digital fingerprint from users interaction with system and validating whether the user is impersonator or a legitimate user. The beauty of this technology is that behavioural fingerprints are highly detailed and virtually impossible to imitate. This chapter will include the methodology to implement such

systems.

Chapter 9, Speeding Things up with GPU

The machine learning includes data intensive computation tasks as training the system or a neural net takes massive amount of computer processing that spans upto days or weeks depending upon its complexity. Using GPU, developers can build more sophisticated solutions which in turn leads to intelligent next generation systems.
Speed comparison of vector multiplication example with and without GPU

Chapter 10, Change the Game with TensoFlow

TensorFlow is open source software library by google that can make use of either CPU's or GPU’s, or a mix of them. It allows us to express arbitrary computations as data flows or graphs, it can be used to design artificial neural network to detect patterns and correlations just like humans learns and reasons things.
TensorFlow provides high level python interface and efficiency of C in an easy to understand framework.
We will use TensorFlow to re implement some of the above examples to compare and contrast its results.

Chapter 11, Financial Frauds and How Deep Learning can Mitigate Them

Financial frauds like credit card fraud, mortgage or loan fraud are taking places everywhere and with the access to internet based financial services the rate is increasing tremendously. Fraud in this aspect not necessarily mean people being devious like huge credit card frauds on payment gateways, it also includes where somebody is lying bits about their previous salary or employment history. So how do we catch this kind of frauds ?Thats where deep learning jumps in, we can create models and train the system from previous fraudulent events so that they can be mitigated in future.

Basics of Machine Learning in Cyber Security

The goal of this chapter is to introduce Cyber Security professionals to the basics of Machine Learning. We introduce to the readers the overall architecture for running Machine Learning modules and go through in great details the different subtopics in the machine learning landscape.

The are many books of machine learning that are dealing with practical use-cases but very few address the cyber security and the different stages of the threat life cycle. This book is aimed for cyber security professionals who are looking forward to detect threat by applying machine learning and predictive analytics.

In this chapter we go through the basics of machine learning. The primary areas that we cover are:

- Definitions of Machine Learning and use-cases
- Delving with machine learning the cyber security world
- Different Types of Machine Learning Systems
- Different Data Preparation Techniques
- Machine Learning Architecture
- A deeper dive to statistical models and Machine Learning Models
- Model Tuning to ensure model performance and accuracy
- Machine Learning Tools

What is Machine Learning?

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

– Tom M. Mitchell

Machine Learning is the branch of science that enables computers to learn, to adapt, to extrapolate patterns and communicate with each other without explicitly being programmed to do so. The term dates back to history, in 1959 when it was first coined by Arthur Samuel at the IBM Artificial Intelligence Labs. Machine learning had its foundation in statistics and now overlaps significantly with data mining and knowledge discovery. In the following chapters we will go through a lot of these concepts using cyber-security as the back drop.

In 1980's machine learning gained much more prominence with success of artificial neural networks. Machine Learning got glorified in the 1990s when researchers started using them to solve day to day life problems. In early 2000s the internet and digitization poured fuel to this wild fire and over the years companies like Google, Amazon, Facebook, Netflix started leveraging machine learning to improve human- computer interactions even further. Voice recognition and Face recognition systems have become our goto technologies. More recently artificially intelligent home automation products, self driving cars, robot butlers have sealed the deal.

The field of cybersecurity however at this same period saw several massive cyber attacks and data breaches. These are regular attacks as well as state sponsored attacks. Cyber attacks have gone so big that criminals these days are not content with regular impersonations and account take overs, they target massive industrial security vulnerabilities and try to achieve maximum ROI(return of investment) from a single attack. Several fortune 500 companies have fallen prey to sophisticated cyber attacks, spear fishing attacks, zero day vulnerabilities and so on. Attacks on IOT(Internet of Things) devices and cloud have gained momentum. These cyber breaches seemed to outsmart human SOC(security operations center) analysts and machine learning methods are needed to complement human effort. More and more threat detection systems are now dependent on these advanced intelligent techniques and are slowly moving away from the signature based detectors typically used in Security Information and Event Management (SIEM).

Problems that Machine Learning solves

The following table presents some of the common problems that machine learning solves:

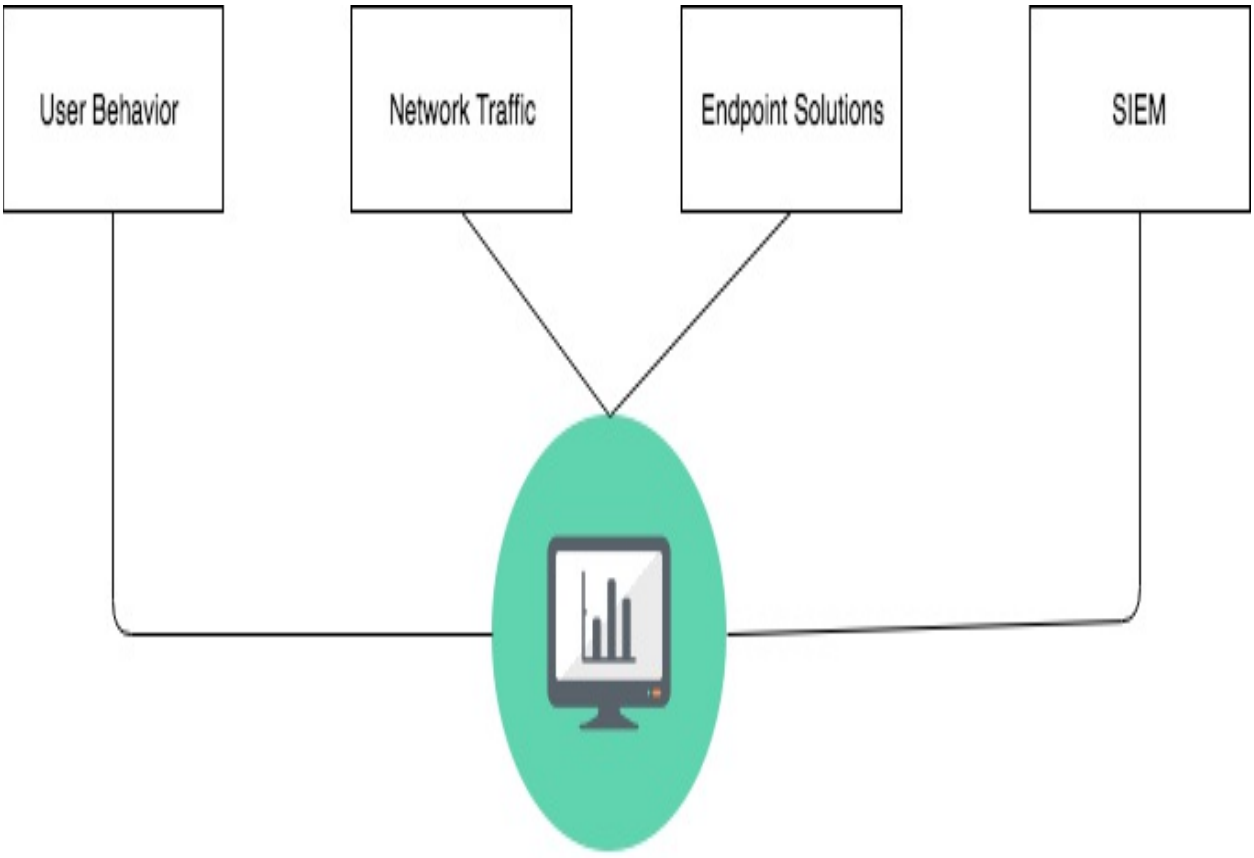
Use-case Domain	Description
Face Recognition	Face recognition systems can identify people from digital images by recognizing facial features. These are similar to biometrics and extensively used security systems like the use of face recognition technology to unlock phone. Such systems use 3-dimensional recognition and skin texture analysis to verify faces.
Fake News Detection	Fake news is rampant specially since the 2016 US Presidential elections. To stop such yellow journalism and the turmoil created by fake news, detectors were introduced to separate fake news from legitimate news. The detectors use semantic and stylistic patterns of the text in the article, the source of article and so on to segregate fake from legit.
Sentiment Analysis	Understanding the overall positivity or negativity of a document is important as opinion is an influential parameter while making a decision. Sentiment Analysis systems perform 'opinion mining' to understand the mood and attitude of the customer.
Recommender Systems	These are systems that are able to assess the choice of a customer based on the personal history of previous choices made by the customer. Another determining factor that influences such systems choices made by other similar customers. Such recommender systems are extremely popular and heavily used by industries to sell movies, products, insurances and so on. Recommender systems in a way decided the go to market strategies for the company based on cumulative like or dislike.
Fraud Detection Systems	Fraud Detection systems are created to for risk mitigation and safe fraud the customer interest. Such systems detect outliers in transactions and raise flags by measuring the anomaly coefficients.
Language Translators	Language translators are intelligent systems that are able to translate not just word to word but whole paragraphs at a time. Natural language translators use contextual information from multilingual documents and are able to make these translations
Chat Bots	Intelligent chatbots are systems that enhance customer experience by providing auto responses when an human customer service agent cannot respond. However their activity is not just limited to being a virtual assistant. They have sentiment analysis capabilities and also able to make recommendations.

Why use Machine Learning in Cyber Security?

Legacy based threat detection system used heuristics and static signatures on large amount of data logs to detect threat and anomalies. However this meant that analysts must be aware of what normal data logs should look like. The process included data being ingested and processed through the traditional Extraction, Transformation and Load(ETL) phase. The transformed data is read by machines and analyzed by analysts who create signatures. The signatures are then evaluated by passing more data. An error in evaluation meant rewriting the rules. Signature based threat detection techniques though well understood are not robust since signatures need to be created on the go for larger volumes of data.

Current day cyber security solutions

Today signature based systems are being gradually replaced by intelligent cyber security agents. Machine learning products are aggressive in identifying new malware, zero day attacks and advanced persistent threats. Insight from the immense amount of log data is being aggregated by log correlation methods. Endpoint solutions have been super active in identifying peripheral attacks. New machine learning driven cyber security products have been proactive to strengthen the container systems like virtual machine. The following diagram gives a brief overview of some of the machine learning solutions in cyber-security:



Machine Learning Solutions in Cyber Security

In general, machine learning products are created to predict attacks before they occur but given the sophisticated nature of these attacks preventive measures often fail. In such cases machine learning often helps to remediate in other ways, like recognizing the attack at its initial stages and prevent it from spreading across the entire organization.

More and more cyber security companies are relying on *advanced analytics* such as user behavior analytics and predictive analytics to identify advanced persistent threats early on in the threat life cycle. These methods have been successful in preventing data leakage of personally identifiable information(PAI) and insider threats. But prescriptive analytics is another advanced

machine learning solution that is worth mentioning in the cyber security perspective. Unlike predictive analytics which predicts threat by comparing current threat logs with historic threat logs, prescriptive analytics is a more reactive process. Prescriptive analytics deals with situations where a cyber attack is already in play. It analyzes data at this stage to suggest what reactive measure could best fit the situation to keep the loss of information to a minimal.

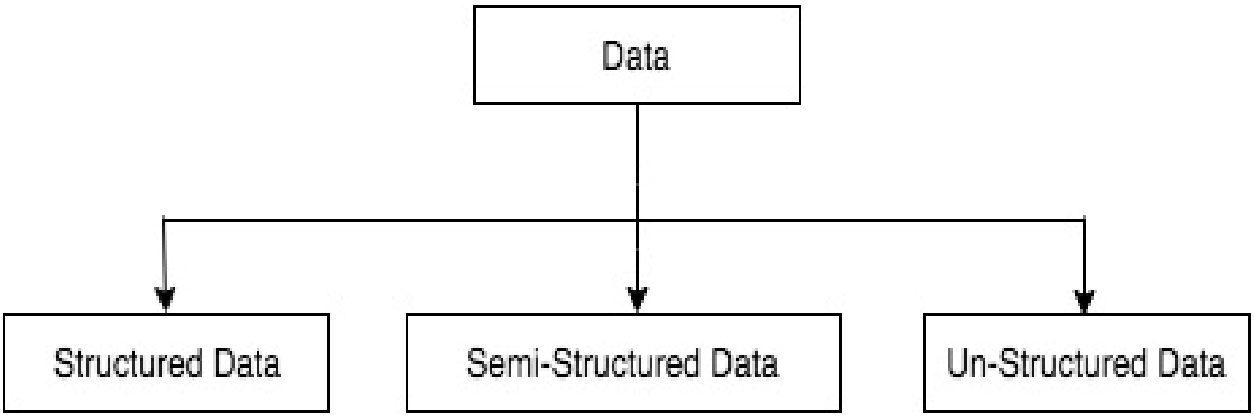
Machine learning however has a down side in cyber security. Since alerts generated need to be tested by human SOC analysts, generating too many false alerts could cause alert fatigue. To prevent this issue of false positives, cyber security solutions also get insights from SIEM signals. The signals from SIEM systems are compared with the advanced analytics signals so that the system does not produce duplicate signals. Thus machine learning solution in the field of cyber security products learn from environment to keep false signals to a minimum.

Data in Machine Learning

Data is the fuel that drives the machine learning engine. Data when fed to Machine Learning systems help in detecting patterns and mining the data. This data can be in any form and come in frequency from any source.

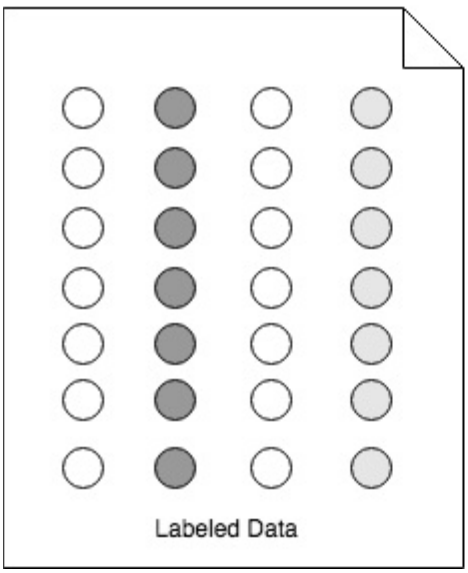
Structured v/s Unstructured Data

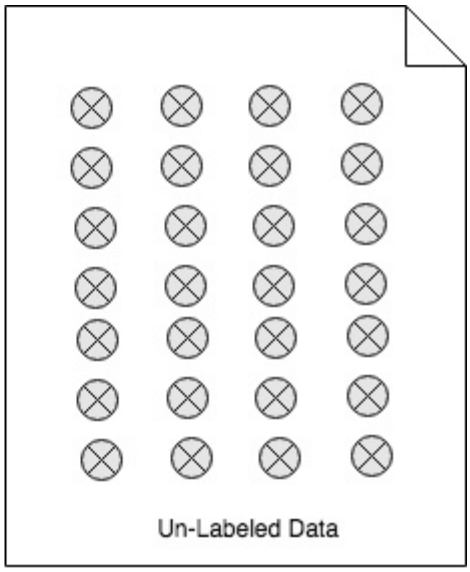
Depending on the source of origination of data and the use-case in hand, data can be either be structured data that is they can be easily mapped to identifiable column headers, or data can be unstructured that is they cannot be mapped to any identifiable data model. A mix of unstructured and structured is called semi structured data. We will discuss later in the chapter the machine learning approaches to handle these two type of data is different.



Labeled v/s Unlabeled Data

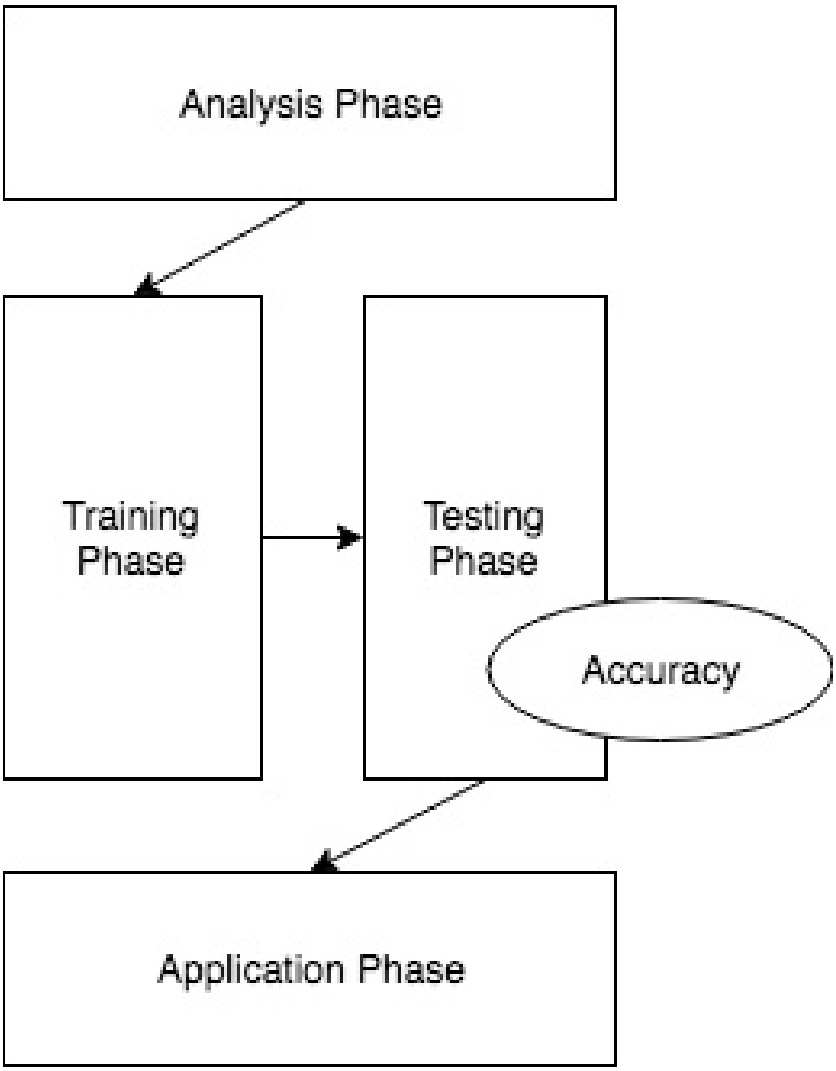
Data can also be categorized into labelled and unlabelled data. Data that have been manually tagged with headers and meaning is called labelled. Data that has not been tagged is called unlabeled data. Both labeled and unlabelled data are fed to the above machine learning phases. In the training phase the ratio of labelled to unlabelled is 60-40 and 40-60 in the testing phase. Unlabelled data is transformed to labeled data in the testing phase.





Machine Learning Phases

The general approach to solving machine learning consists of a series of phases. These phases are consistent no matter what the source of data is. That is be it structured or unstructured the stages required to tackle any kind of data are as follows:



1. **The Analysis Phase :** In this phase the ingested data is analyzed to detect patterns in the data which help create explicit features or parameters that can be used to train the model.
2. **The Training Phase :** Data parameters generated in the previous phases is used to create machine learning models in this phase. The training phase is iterative process where the data incrementally helps to improve the quality of prediction.
3. **The Testing Phase :** Machine learning models created in the training phase are tested with more data and the model's performance is assessed. In this stage we test with data that has not been used in previous phase. Model evaluation at this phase may or may not require

parameter training.

4. **The Application Phase :** The tuned models are finally fed with real world data at this phase. At this stage the model is deployed in the production environment.

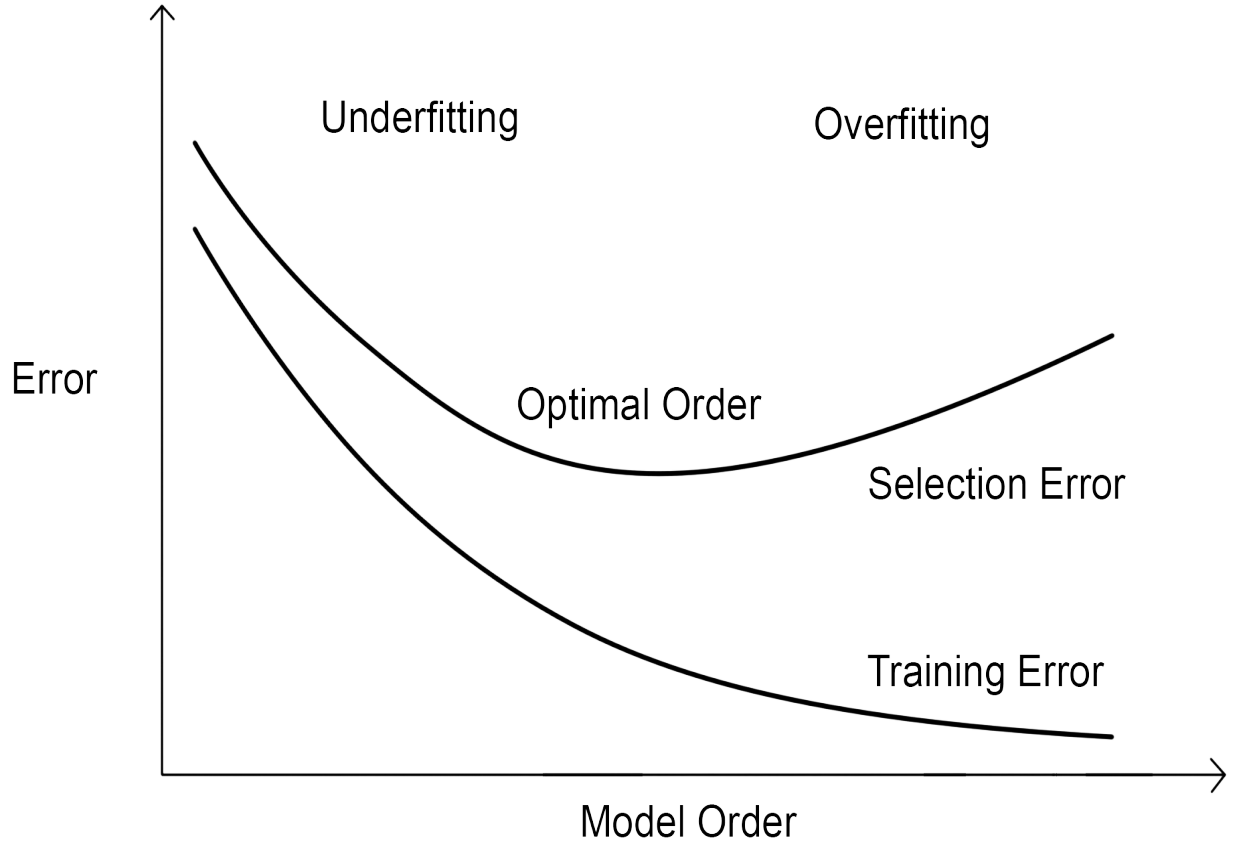
Inconsistencies in Data

In the training phase a machine learning model may or may not generalize perfectly. This is due to the inconsistencies that we need to be aware of.

Over fitting

"The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably" - Oxford Dictionary

Over fitting is the phenomenon in which the system is too fitted to the training data. The system produces a negative bias when treated with new data. In other words the models perform badly. Often this because we feed only labeled data to our model. Hence we need both labelled and unlabelled data are required to train a Machine Learning system.



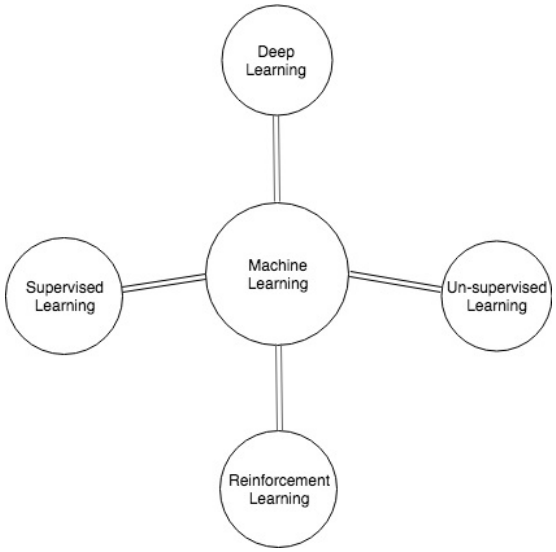
Under-fitting

Under-fitting is another scenario where model performs badly. This is a phenomenon where the performance of the model is affected because the model is not well trained. Such systems have trouble in generalizing to new data.

For ideal model performance both overfitting and under-fitting can be prevented by performing some common machine learning procedures like cross validation of the data, data pruning and regularization of the data. We will go through these with much more details in the following chapters after we get more acquainted with the machine learning models. The graph above shows that to prevent any model errors we need to select data in the optimal order.

Different Types of Machine Learning Algorithms

In this section we will be discussing the different types of machine learning systems and the most commonly used algorithms with special emphasis to the ones that are more popular in the field of cyber security.



Machine Learning Systems can be broadly categorized to two types supervised approaches and unsupervised approaches based on the types of learning they provide.

Supervised Learning Algorithms

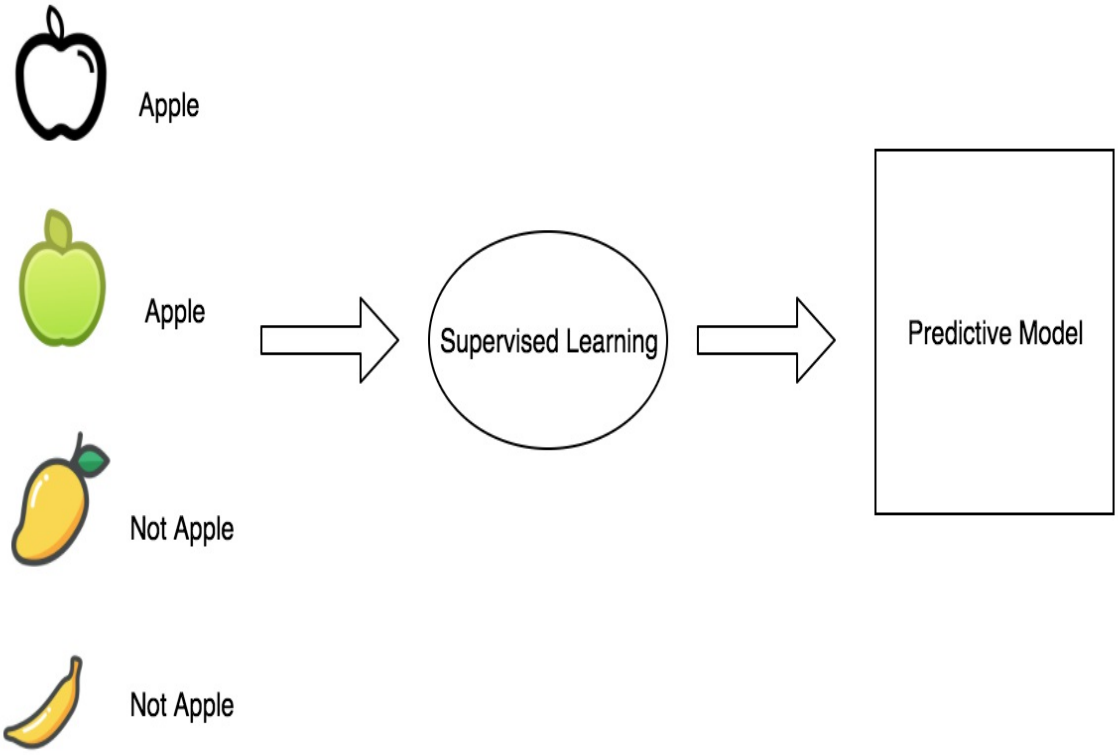
Supervised Learning is the learning technique where a known data set is used to classify or predict with data in hand. Supervised learning methods learn from labelled data and then uses the insight to make decisions on the testing data.

Supervised Learning has several sub categories of learning like:

- Semi-Supervised Learning: This is the type of learning where the initial training data is incomplete. In order words in this type of learning both labeled and unlabeled are used in the training phase.
- Active Learning: In this the algorithm the Machine Learning system by active queries made to the user and learns on the go. This is a specialized case of supervised learning.

Some popular examples of the supervised learning are:

1. Face Recognition : Face recognizers use supervised approaches to identify new faces. Face recognizers extract information from a bunch of facial images that are provided to it during the training phase. It uses insight gained after training to detect new faces.
2. Spam Detect : Supervised learning helps distinguishing spam emails in the inbox by separating them from legitimate emails also known as "ham" emails. During this process the training data enables the learning which helps such systems to send the ham emails to the inbox and spam emails to the spam folder.

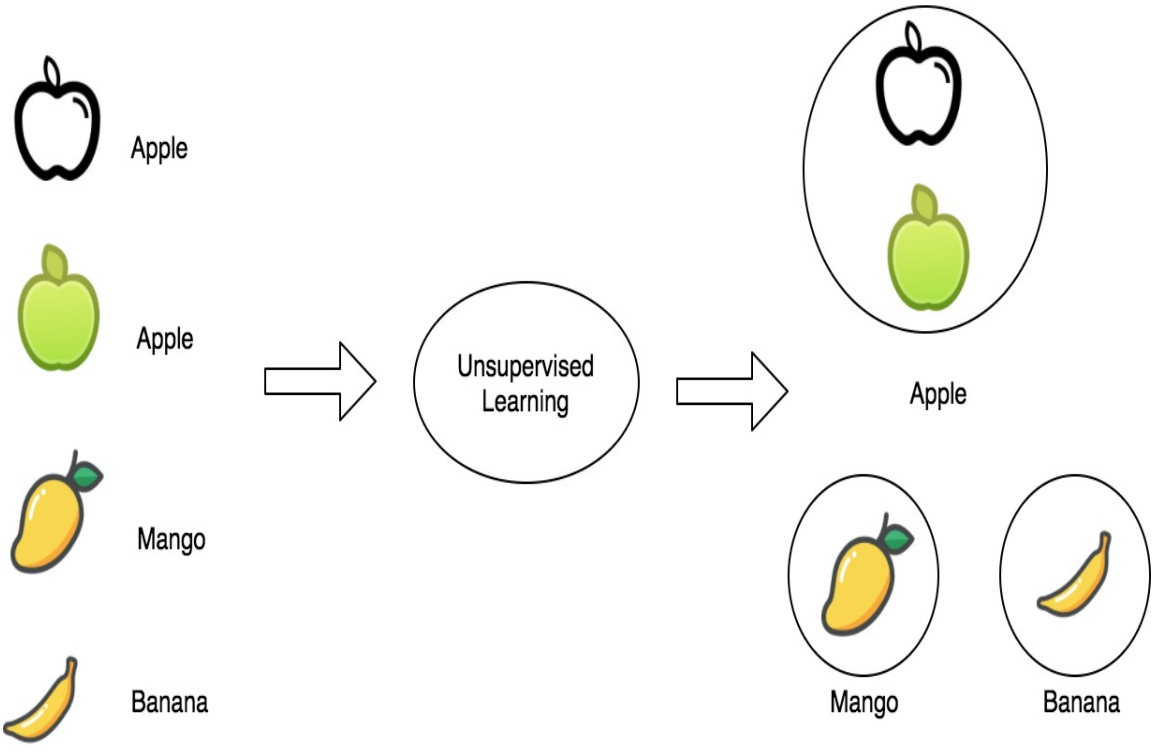

Apple
Apple
Not Apple
Not Apple
Supervised Learning
Predictive Model

Unsupervised Learning Algorithms

Unsupervised learning technique is the type of learning where the initial data is not labelled. Insights are drawn by processing data who structure is not known before hand. These are more complex processes since the system learns by itself without any intervention.

Some practical examples of unsupervised learning techniques are:

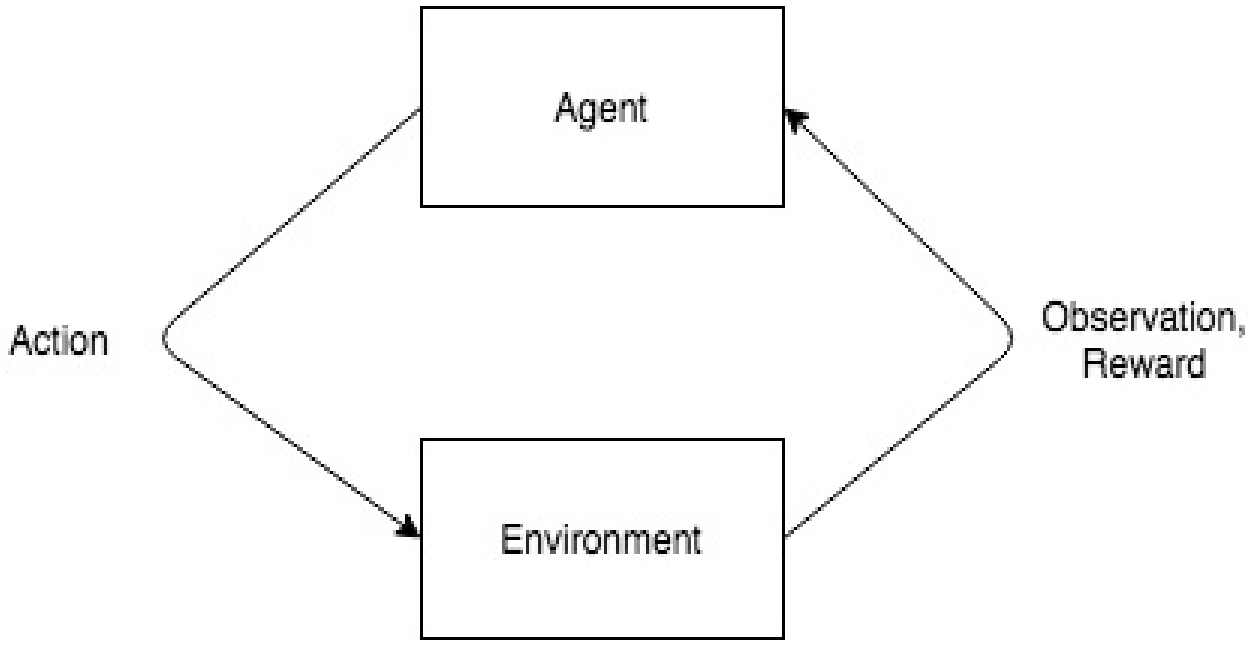
1. User Behavior Analysis: Behavior analytics uses unlabelled data about different human traits and human interactions. This data is then used to put each individual to different groups based on their behavior patterns
2. Market Basket Analysis : This is another example where unsupervised learning helps identify the likeliness that certain items always appear together . An example of such a analysis is the shopping cart analysis where chips, dips and beer are likely to be found together in the basket.



Reinforcement Learning

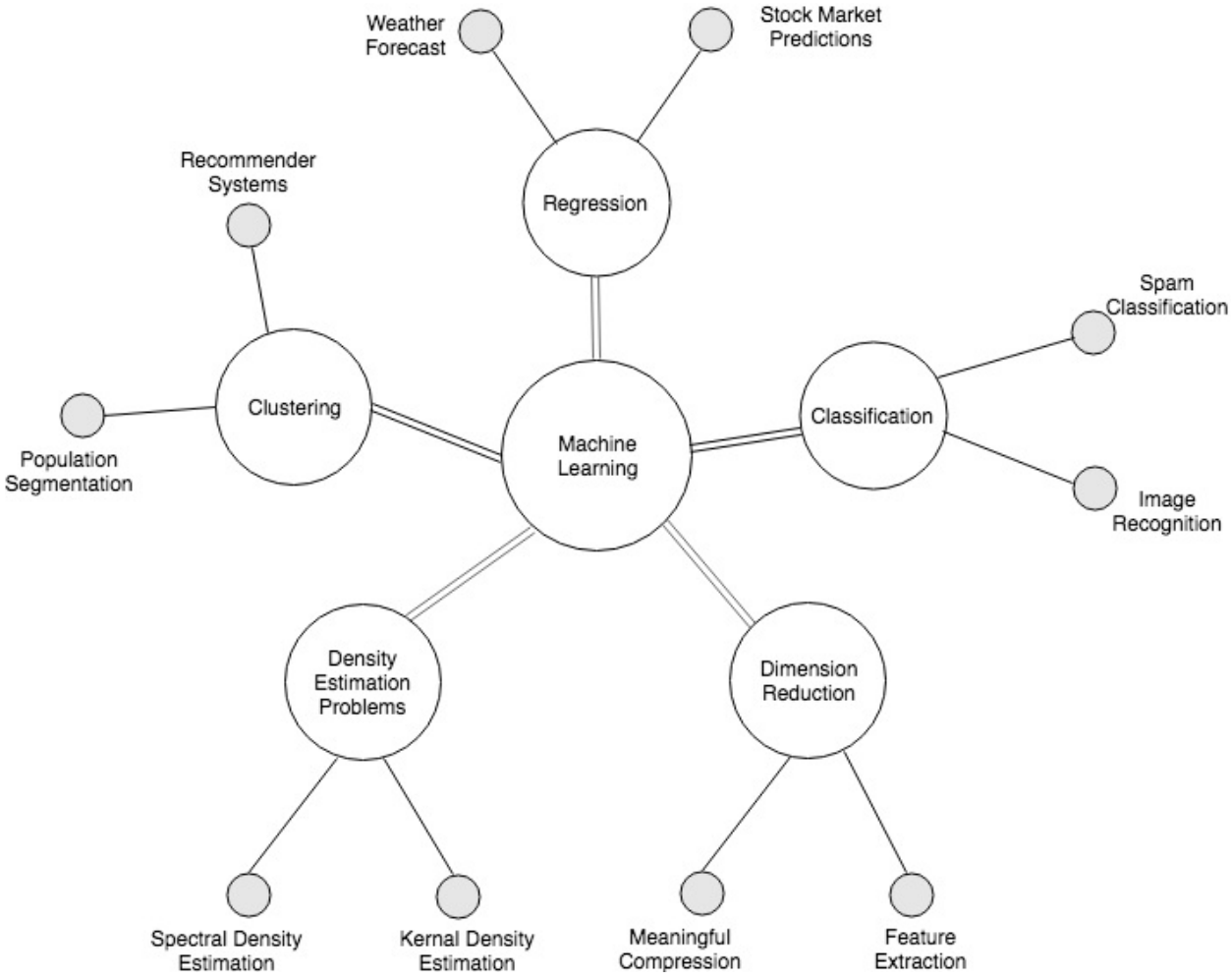
Reinforcement learning is a type of dynamic programming where the software learns from its environment to produce an output that will maximize the reward. Here the software require no external agent but learns from the surrounding processes in the environment.

1. Self Driving Cars: Self Driving cars exhibit autonomous motion by learning from the environment. The robust vision technologies in such a system are able to adapt from surrounding traffic conditions. this when amalgamated with complex software and hardware movements make it possible to navigate through the traffic.
2. Intelligent Gaming Programs: Deep minds's artificially intelligent "GO" program has been successful in learning a number of games in a matter of hours. Such systems use reinforcement learning in the background to quickly adapt game moves. The "GO" was able to beat world known AI chess agent Stockfish with a just four hours of training.



Another Categorization of Machine Learning

Machine Learning Techniques can also be categorized by the type of problem they solve like the Classification, clustering, regression, dimensionality reduction and density estimation techniques. Below we briefly discuss the definitions and examples of these systems. In the next chapter we will be delving with details and its implementation with respect to cyber security problems.

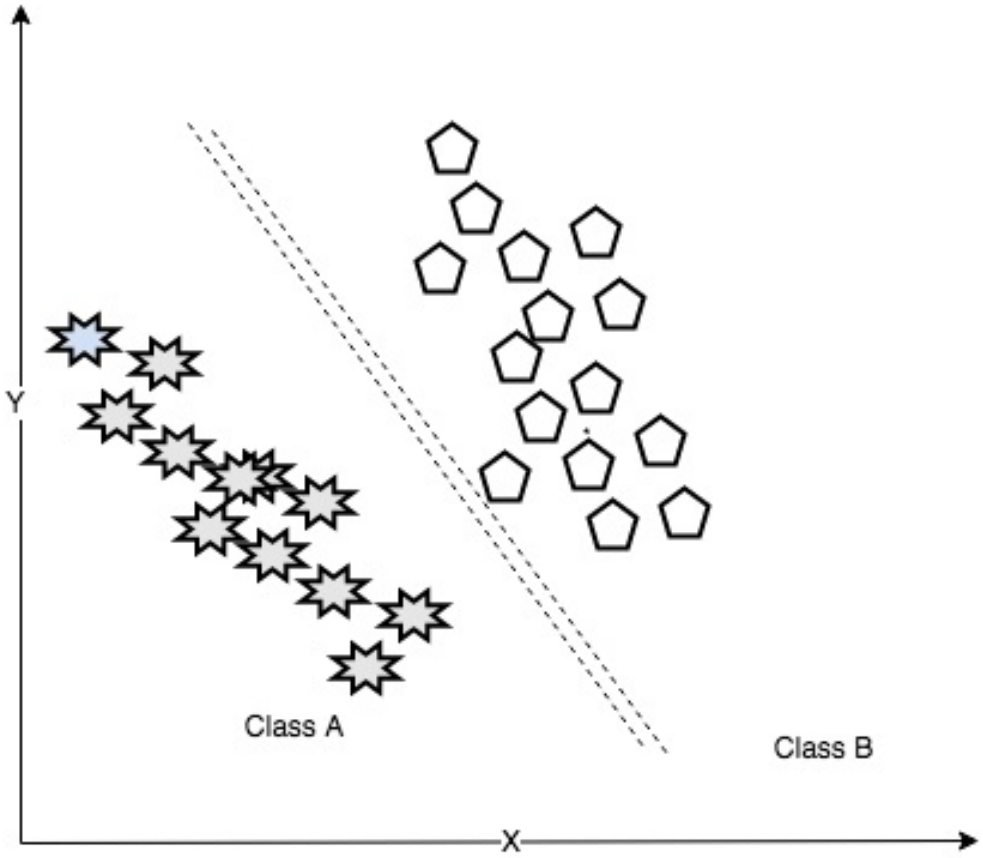


Machine Learning Algorithms

Classification Problems

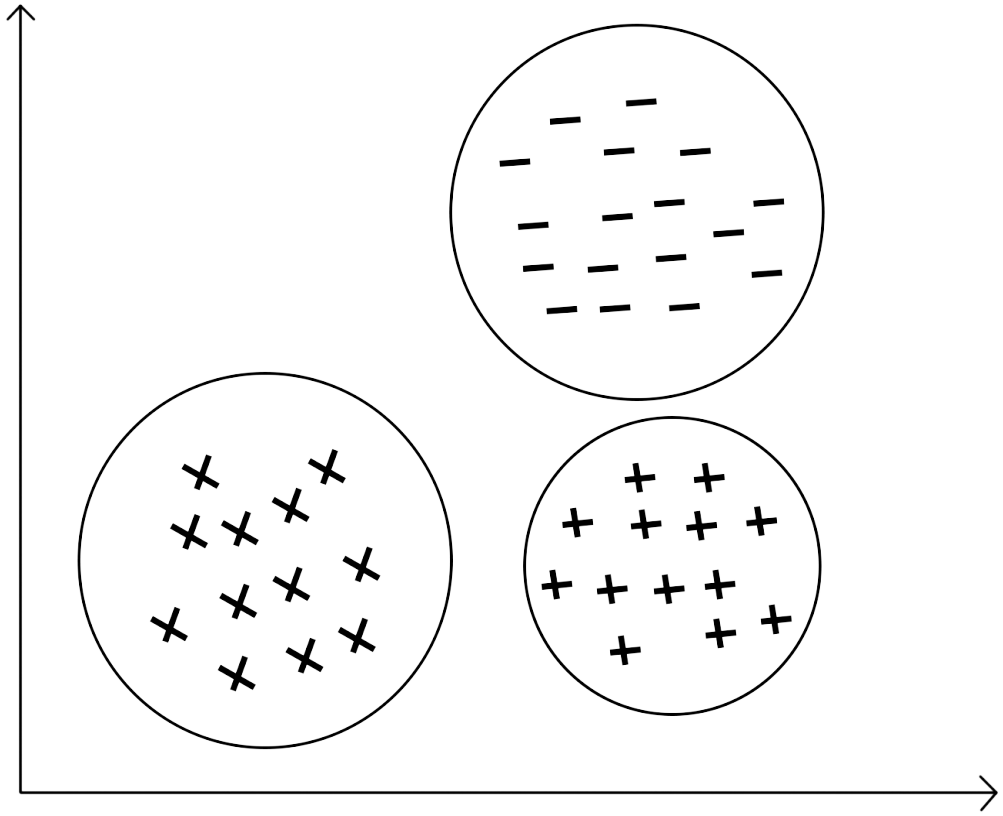
Classification is the process of dividing data into multiple classes. Unknown data is ingested and divided into categories based on characteristics or features. Classification problem is an instance of supervised learning since the training data is labeled here.

Web data classification is a classic example of this type of learning where web contents get categorized with models to their respective type based on their textual content like news, social media, advertisements and so on.



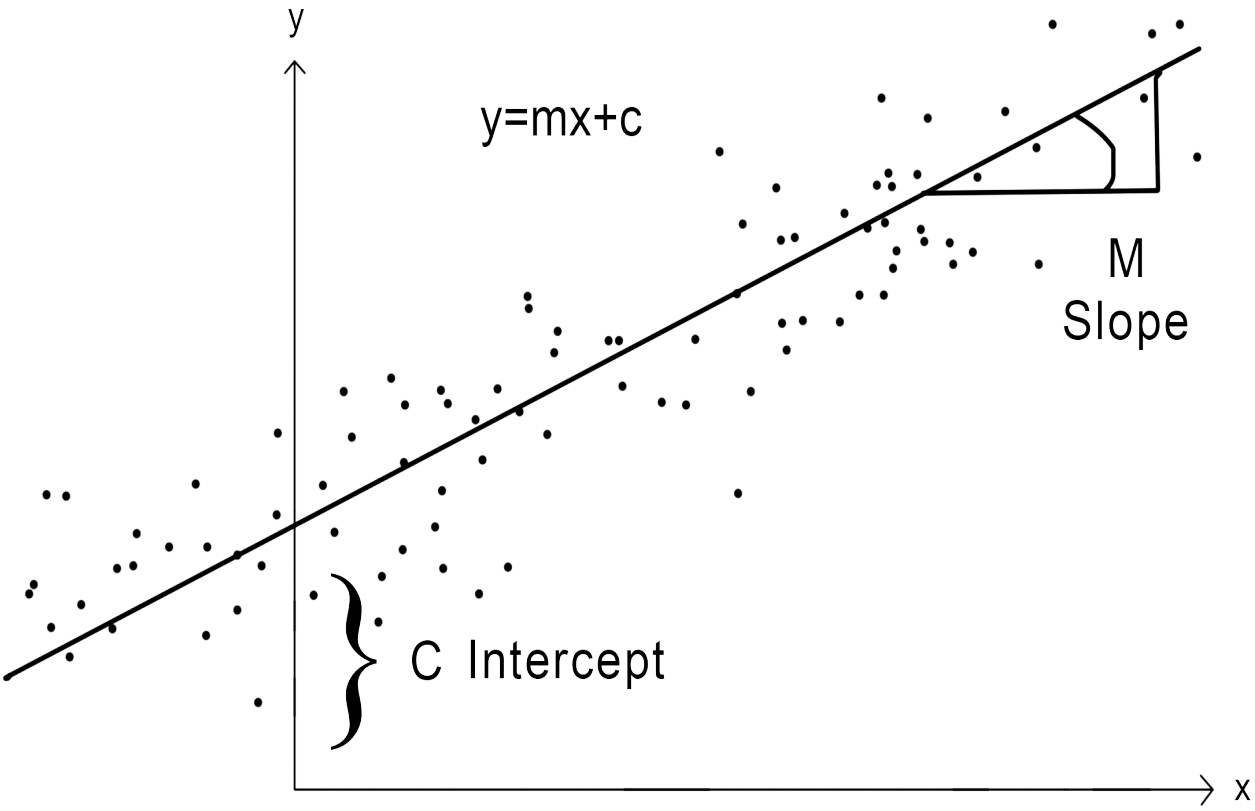
Clustering Problems

Clustering is the process of grouping data and putting similar data into the same group. Clustering techniques use series of data parameters and go through several iterations before they can group the data. These techniques are most popular in the fields of information retrieval and pattern recognition. Clustering techniques are also popularly used in the demographic analysis of the population.



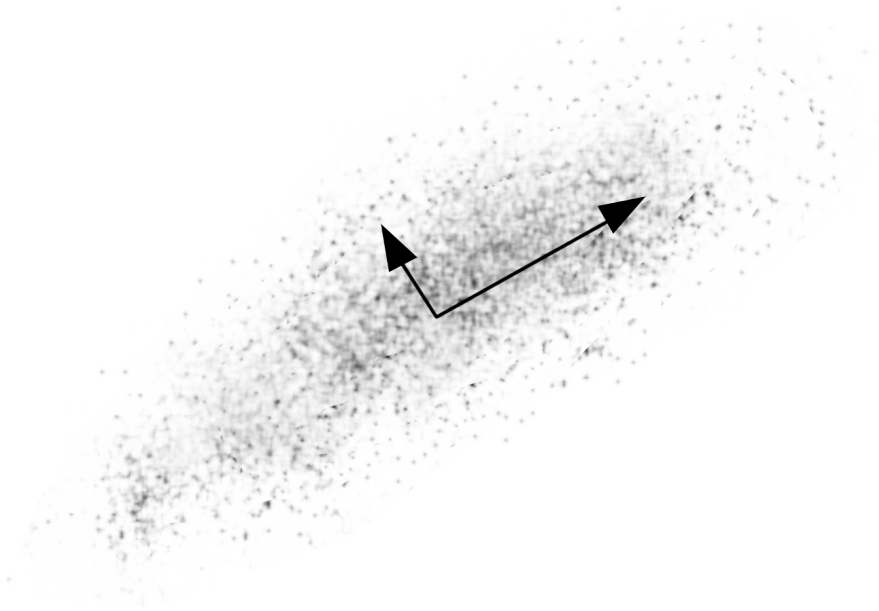
Regression Problems

Regression are statistical process of analyzing data that help with both classification and prediction with data. In regression the relationship between two variables present in the data population is estimated by analyzing multiple independent and dependent variables. Regression can be of many types like linear regression, logistic regression, polynomial regression, lasso regression and so on. An interesting usecase with regression analysis are fraud detection systems. Regressions are also used in stock market analysis and prediction.



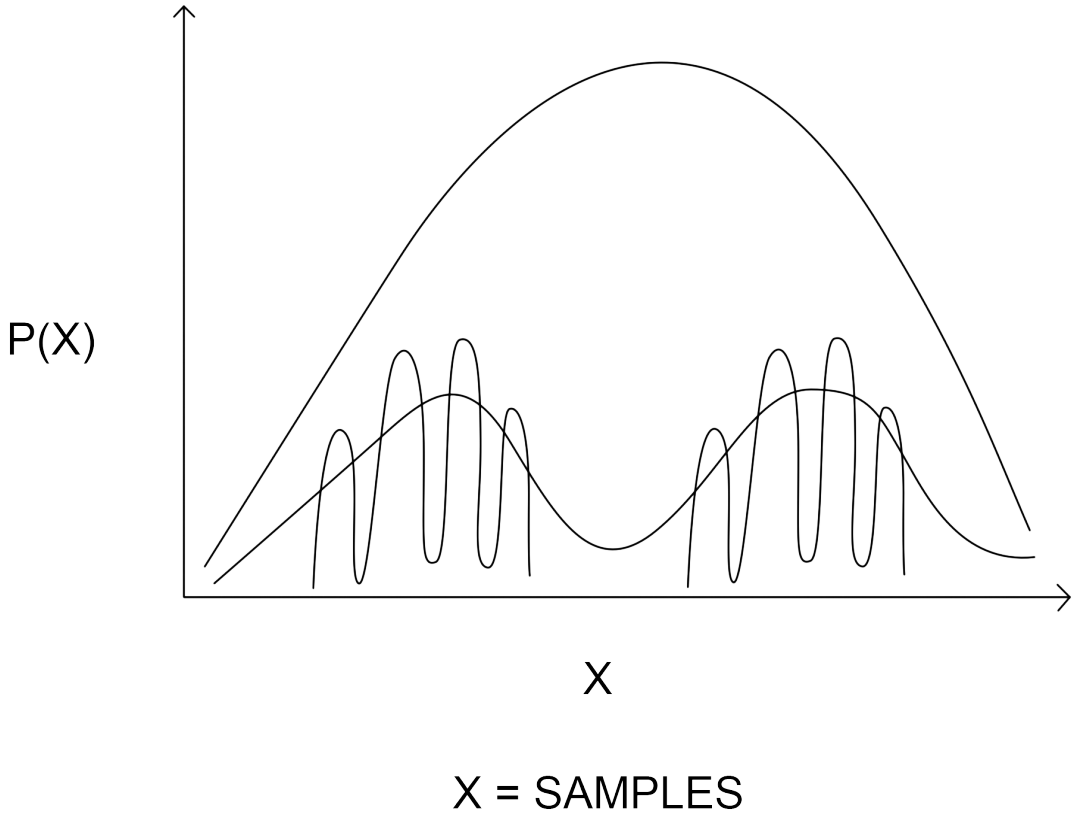
Dimensionality Reduction Problems

Dimensionality reduction are machine learning techniques where high dimensional data with multiple variables get represented with its principle variables without loosing any vital data. Dimensionality reduction techniques are often applied on network packet data to make the volume of data sizable. These are also used in the process for feature extraction where it is impossible to model with high dimensional data.



Density Estimation Problems

Density Estimation Problems are statistical learning methods used in machine learning estimations from dense data that are otherwise unobservable. Technically density estimation is technique of computing probability of the density function. Density estimation can be applied on path parametric and non parametric data. Medical analysis often use these techniques for identifying symptoms related to diseases from a very huge population.



Deep Learning

Deep Learning is the form of machine learning where systems learn by examples. These are more advanced form of machine learning. Deep learning is the study of deep neural networks and require much larger data set. Today deep learning is the most sought after techniques in technology. Some popular examples of deep learning applications include self driving card, smart speakers, home-pods and so on.

Algorithms in Machine Learning

So far we dealt with different machine learning systems, in this section we discuss the algorithms that drive them. The algorithms discussed her fall under one or many groups of machine learning that we have already done.

Support Vector Machines

Support Vector Machine or SVMs are supervised learning algorithms used in both linear and non linear classification. SVMs operate by creating an optimal hyperplane in high dimensional space. The separations created by this hyperplane is called Class. SVMS need very little tuning once trained. They are used in high performing systems because of the reliability they have to offer.

SVM are also used in regression analysis is used in ranking and categorization.

Bayesian Networks

Bayesian Network (BN) are probabilistic models that are primarily used for prediction and decision making. these are belief network that uses principles of probability theory along with statistics.BN uses directed acyclic graph (DAG) to represent the relationship of variables and any other corresponding dependencies.

Decision Trees

Decision tree learning is a predictive machine learning technique that uses decision trees. Decision trees make use of decision analysis and predicts the value of the target. Decision trees are simple implementations of classification problems and popular in operations research. Decisions are made by the output value predicted by the conditional variable.

Random Forests

Random Forests are extensions of decision tree learning. Here several decisions trees are collectively used to make prediction. Since this is an ensemble they are stable and reliable. random forests can go in treat depth to make irregular decisions. A popular use-case for random forest is quality assessment of text documents.

Hierarchical Algorithms

Hierarchical algorithms are a form of clustering algorithm. they are sometimes referred as the Hierarchical Clustering Algorithm(HCA). HCA can either be bottom up or agglomerative or they may be top down or divisive. In the agglomerative approach the first iteration comprises of forming its own cluster and gradually smaller clusters are merged to move up the hierarchy. The top down divisive approach start with a single cluster that is recursively broken down into multiple clusters.

Genetic Algorithms

Genetic Algorithms are meta-heuristic algorithms used in 'constrained and unconstrained optimization problems'. They mimic the physiological evolution process of humans and uses the insight to solve problems. Genetic algorithms are known to outperform some traditional machine learning and search algorithms because they can withstand noise or changes in input pattern.

Similarity Algorithms

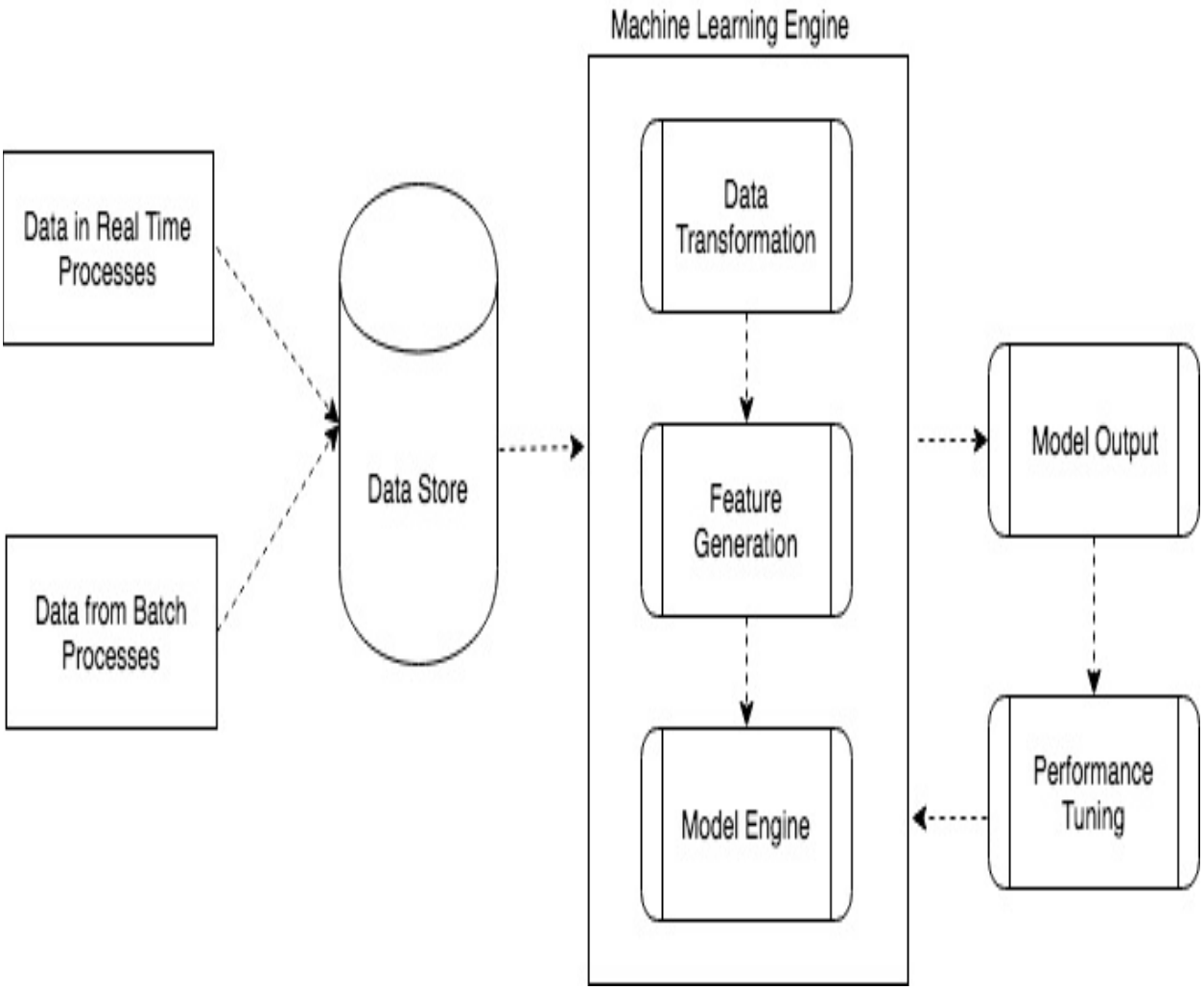
Similarity Algorithm are prevalently used in the field of text mining.Cosine similarity is a popular algorithm primarily used to compare the similarity between documents. The inner product space of two vectors identifies the amount of similarity between two documents. Similarity Algorithms are used in authorship and plagiarism detection techniques.

Artificial Neural Networks

Artificial Neural Networks (ANN) are intelligent computing systems that mimic the human nervous system. ANN comprises of multiple multiple nodes both input and output. These input and output nodes are connected by a layer of hidden nodes. The comples relationship between input layers helps in finding pattern and in problem solving like the human body does.

The Machine Learning Architecture

A typical machine learning system comprises of a pipeline of process that happen in sequence for any type of machine learning systems irrespective of the industry.Figure () shows a typical machine leaning system and the sub processes involved.



Data Ingest

Data is ingested from different sources like real time systems like IOTS(CCTV cameras,), streaming media data and transaction logs. Data that is ingested can also be data from batch processes or non interactive processes like linux cron jobs, windows scheduler jobs and so on. Single feed data like raw text data, log files, process data dump are also taken in by the Data Stores. Data from ERP(Enterprise Resource Planning), CRM(Customer Relationship Management and operational systems are also ingested. Below we analyze some data ingestors that are used in continuous, real time or batched data ingestion.

- Amazon Kinesis: This is a cost effective data ingestor from the house of Amazon. Kinesis enables terabytes of real time data to be stored per hour from different data sources. The Kinesis Client Library (KCL) helps to build applications on the streaming data and further feed to other Amazon services like the Amazon S3, Redshift and so on.
- Apache Flume: Apache flume is a dependable data collector used for streaming data. Apart from data collection they are fault tolerant and have a reliable architecture . They can also be used aggregation and moving data.
- Apache Kafka: Apache Kafka is another open source message broker used in data collection. This 'high throughput' stream processors works extremely well for creating data pipeline. The cluster centric design helps in creating 'wicked fast' systems.

Some of the other data collectors that are widely used in the industry are Apache Sqoop, Apache Storm, Gobblin, Data Torrent, Syncsort and Cloudera Morphlines.

Data Store

The raw or aggregated data from the data collectors is stored in a data stores like SQL databases, No-SQL databases, Data warehouses and Distributed Systems like HDFS. This data may require some cleaning and preparation if they are unstructured. The file format in which data is recieved varies from database dump, json files, parquet files, avro files and even flat files. For distributed data storage systems the data upon ingestion gets distributed to different file formats.

Some of the popular data stores available for use as per industry standards are :

- RDBMS(Relational Database Management System): RDBMS are legacy storage options and is extremely popular in the data warehouse world. They store data retaining the ACID(Atomicity, Consistency, Isolation, Durability) properties. However they suffer from downsides are storage in volume and velocity.
- MongoDB: MongoDb is a popular noSQL, document oriented database. It has a wide adoption in the cloud computing world. They can handle data in any format like structured, semi- structured and unstructured. With high code push frequency they are extremely agile and flexible. MongoDB is inexpensive compared to other monolithic data storage options.
- Bigtable: This is a scalable no SQL data base from the house of Google. Bigtable is a part of the reliable Google Cloud Platform(GCP). They are 'seamlessly scalable' with a very high throughput. Being a part of GCP enable them to be easily plugged in behind visualization apps like Firebase. This extremely popular among App makers who use it to gather data insights. They are also used for business anaytics.
- AWS Cloud Storage Services: The Amazon AWS is the range of cloud storage services for IOT devices, distributed data storage platforms and databases. AWS data storage services are extremely secure for any cloud computing components.

The Model Engine

A machine learning model engine is responsible for managing the end to end flows involved in making the machine learning framework operational. The process includes data preparation, feature generation, training and testing a model. In the next section we will discuss each of this processes in details.

Data Preparation

Data preparation is the stage where data cleansing is performed to check for consistency and integrity of the data. Once the data is cleansed the data is often formatted and sampled. The data is normalized so that all the data can be measured in the same scale. Data preparation also includes data transformation where the data is either decomposed or aggregated.

Feature Generation

Feature generation is the process where the data in analyzed and we look for patterns and attributes that may influence model results. Features are usually mutually independent and is generated from either raw data or aggregated data. The primary goal of features generation is performing dimensionality reduction and improved performance.

Training

Model Training is the phase in which a machine learning algorithm learns from the data in hand. The learning algorithm detects data patterns, relationships and categories data into classes. The data attributes need to be properly sampled to attain best performance from the models. Usually 70 percent to 80 percent of the data is used in the training phase.

Testing

In the testing phase we validate the model we built in the testing phase. Testing is usually done with 20 percent data. Cross validations methods help determine the model performance. The performance of the model can be tested and tuned. In the next section we discuss the following.

Performance Tuning

Performance tuning and error detection is the most important iteration for a machine learning system as it helps in bettering the performance of the system. Machine learning systems are considered to have optimal performance if the generalized function of the algorithm gives a low generalization error with a high probability. This is conventionally known as Probably Approximately Correct(PAC) theory.

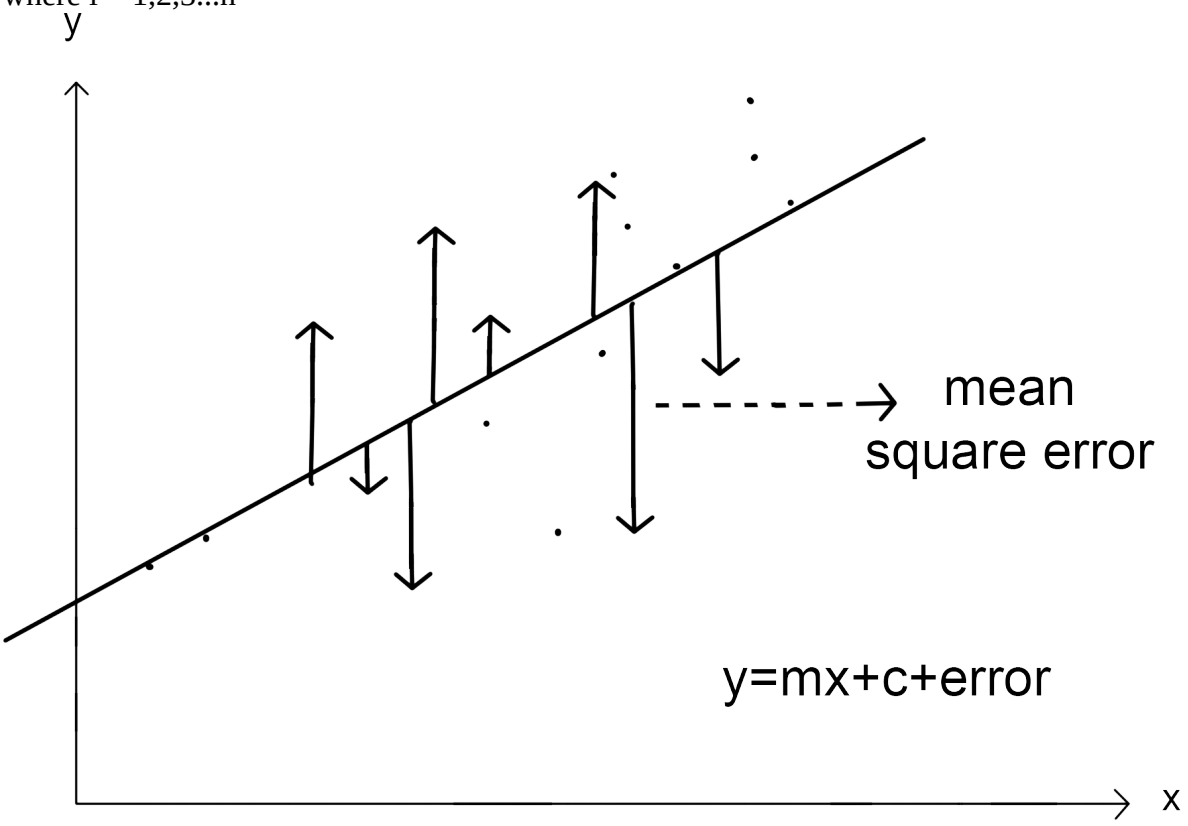
To compute the generalization error i.e. the accuracy of classification or the error in forecast of regression model we use the following metrics:

Mean squared error (MSE)

Imagine for a regression problem we have the line of best fit and we want to measure the distance of each points from the regression line. MSE is the statistical measure that would compute these deviations. MSE computes errors by finding the mean of the squares for each such deviations.

$$MSE = \sum n(Pi - Ai)2/n$$

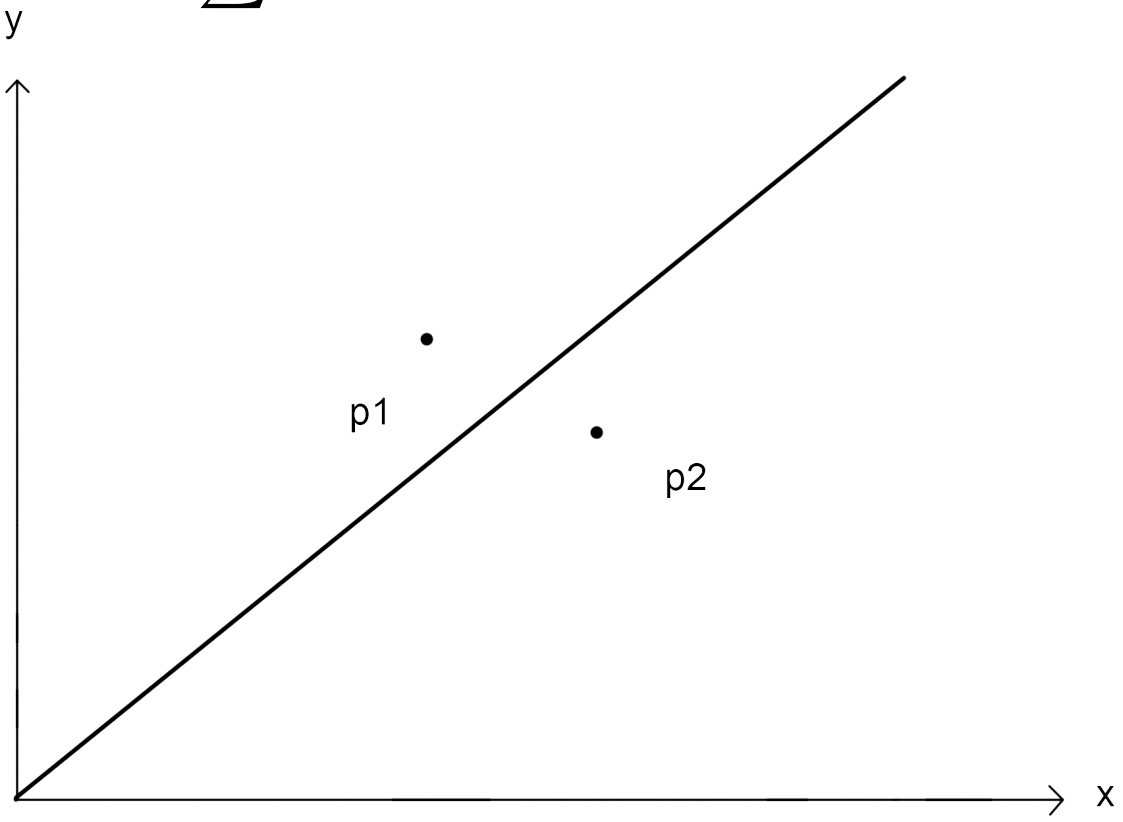
where i = 1,2,3...n



Mean absolute error (MAE)

MAE is another statistical method that helps to measure the distance(error) between two continuous variable. A continuous variable can be defined as a variable that could have infinite number of changing values. Though MAEs are difficult to compute, they are considered as better performing than MSE because it is independent of the square function which have a larger influence on the errors.

$$MAE = \sum ni|Pi - Ai|/n$$



Precision, Recall, Accuracy

Another measure of computing the performance for classification problems is estimating the precision, recall and accuracy of the model.

- Precision is defined as the number of true positives present in the mixture all retrieved instances.

$$Precision(P) = TruePositive/(TruePositive + FalsePositive)$$

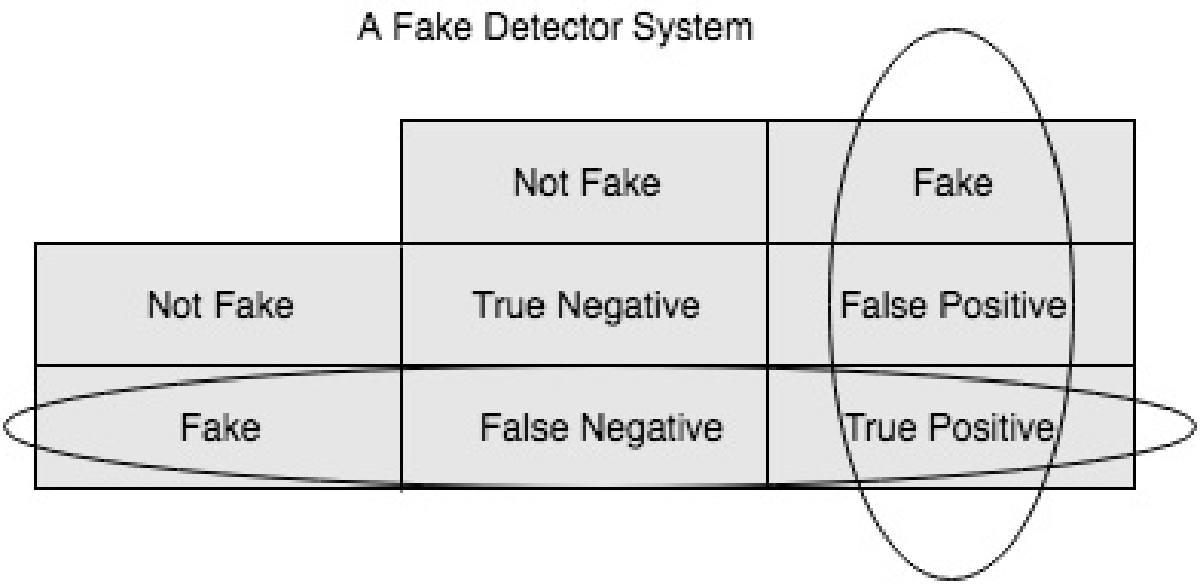
- Recall is number of true positives identified from the total number of true positive present in all relevant documents.

$$Recall = TruePositive/(TruePositive + FalseNegative)$$

- Accuracy: Accuracy Measures the percentage of closeness of the measured value from the standard value.

$$Accuracy = (TruePositive + TrueNegative)/(TruePositive + TrueNegative + FalsePositive + FalseNegative)$$

Fake document detection is real world use-case that could explain the above. For fake news detector systems precision is the number of relevant fake news articles detected from the total number of documents that are detected. Recall on the other hand measures the number of fake news articles that get retrieved from the total number of fake news present. Accuracy measures the correctness with which such a system detects fake news.How can the model performance be improved.



Recall = True Positive / Fake Detector Systems

Precision = True Positive / True Fake Documents

How can the model performance be improved ?

Models with a low degree of accuracy and high generalization error needs improvement to achieve better results. Performance can be improved either by improving the quality of data, switching to different algorithm or tuning the current algorithm performance with ensembles.

Data to Improve Performance

Fetching more data to train a model can lead to improve in performance. A lowered performance can also be due to lack of clean data, hence the data needs to be cleansed, resampled and properly normalized. Revisiting the feature generation can also lead in improved performance. Very often the lack of independent features within a model are causes for its skewed performance.

Switching Machine Learning Algorithms

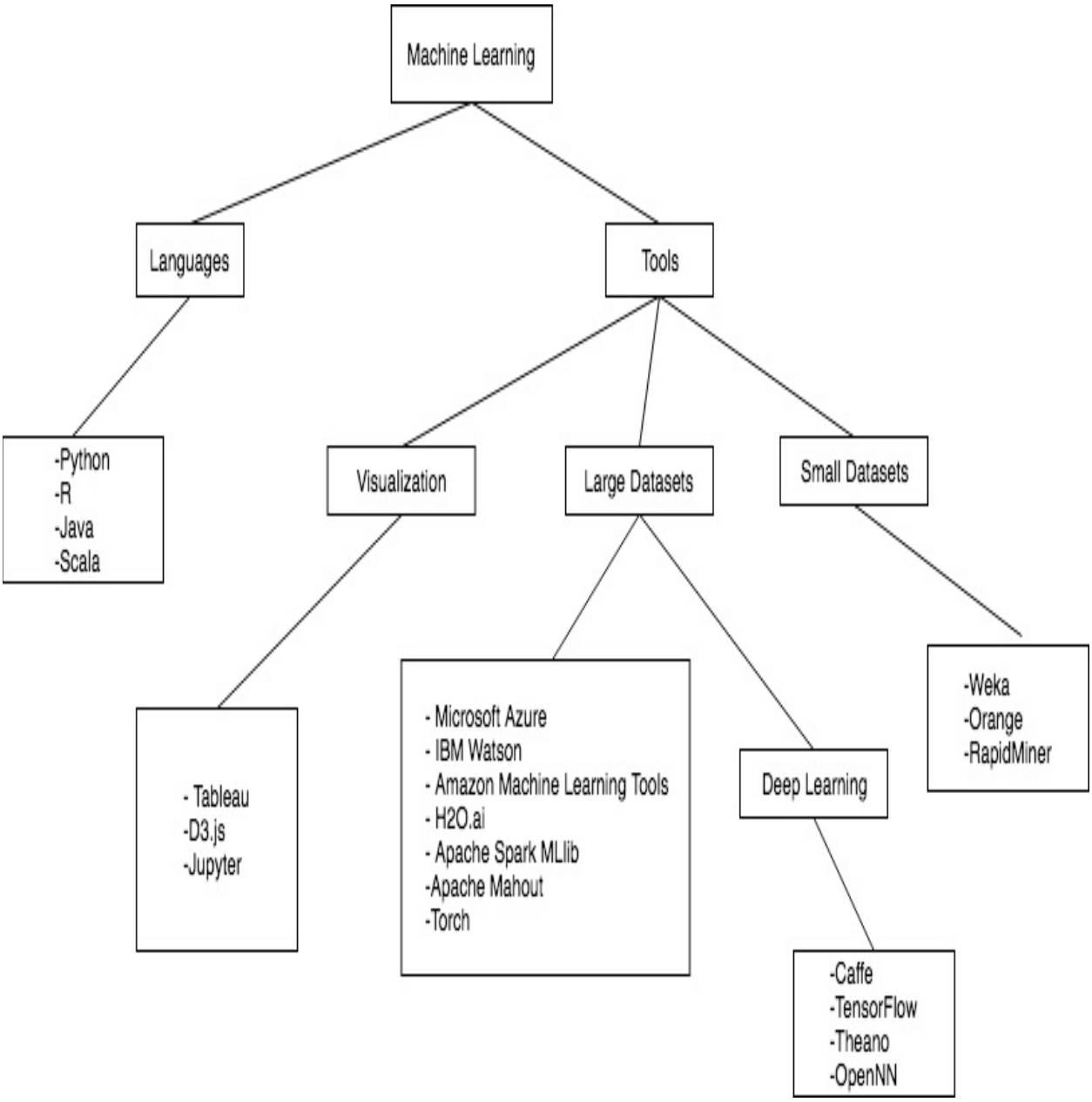
A model performance is often not up to the mark because we have not made a right choice of algorithm. In such scenarios performing a baseline testing with different algorithms helps us make a proper selection. Baseline testing methods include but not limited to k fold cross validations.

Ensemble Learning to Improve Performance

Performance of a model can be improved by ensembling the performance of multiple algorithms. Blending forecasts and data sets can help in making correct predictions. Some of the most complex Artificially Intelligent systems today are a byproduct of such ensembles.

Hands on Machine Learning

We have so far established that machine learning is used heavily in industries and in the field of data driven research. Thus let us go through some machine learning tools that helps creating such machine learning applications with both small or larger scale data. The flow digram shows the various machine learning tools and languages that are currently at our disposal.



Python For Machine Learning

Python is the most preferred language for developing machine learning applications.Though not the fastest , Python is extensively adapted by data scientists because of versatility.

Python supports a wide range of tools and packages that enables machine learning experts to implement changes with much agility. Python being a scripting language is easy to adapt and code in. Python is extensively used for the for graphical user interfaces (GUI) development.

Comparing Python Versions 2.x v/s 3.x

Python 2.x is an older version compared to Python 3.x. Python 3.x was first developed in 2008 while the last python 2.x update came out in 2010. Though it is perfectly okay table application with the 2.x but the it is worthwhile to mention that 2.x not not been developed any further from 2.7.

Almost every machine learning package in use have support for both the 2.x and the 3.x version. However for the purposes of staying up to date we will be using version 3.x in the uses-cases we discuss in this book.

Python installation

Once you have made a decision to install Python2 or Python3, you can download the latest version from Python website in the following url:

```
https://www.python.org/download/releases/
```

On running the file downloaded, python is installed in the following directory unless explicitly mentioned:

For Windows:

```
C:\Python2.x
 C:\Python3.x
```

For Mac:

```
/usr/bin/python
```

For Linux:

```
/usr/bin/python
```

Note: A windows installation would require you to set the environment variables with the correct path

To check the version of python installed you can run the code below:

```
import sys
print ("Python version:{}",format(sys.version))
```

Python Interactive Development Environment(IDE)

The top python IDE commonly used for developing python codes are:

- Spyder
- Rodeo
- Pycharm
- Jupyter

For developmental purposes we would be using IPython Jupyter Notebook due to its user friendly interactive environment. Jupyter allows code transportation and easy mark downs. Jupyter is browser based thus supporting different types of imports, exports and parallel computation.

Jupyter Notebook Installation

To download the Jupyter notebook it is recommended that you:

- First download python, either python 2.x or python 3.x as a prerequisite for Jupyter Notebook installation
- Once python installation is complete download Anaconda from the following link depending on the operating system where the installation is being done. Anaconda is a package/environment manager for python. By default Anaconda comes with 150 packages and another 250 open source package can be installed along with it.

```
https://www.anaconda.com/download/
```

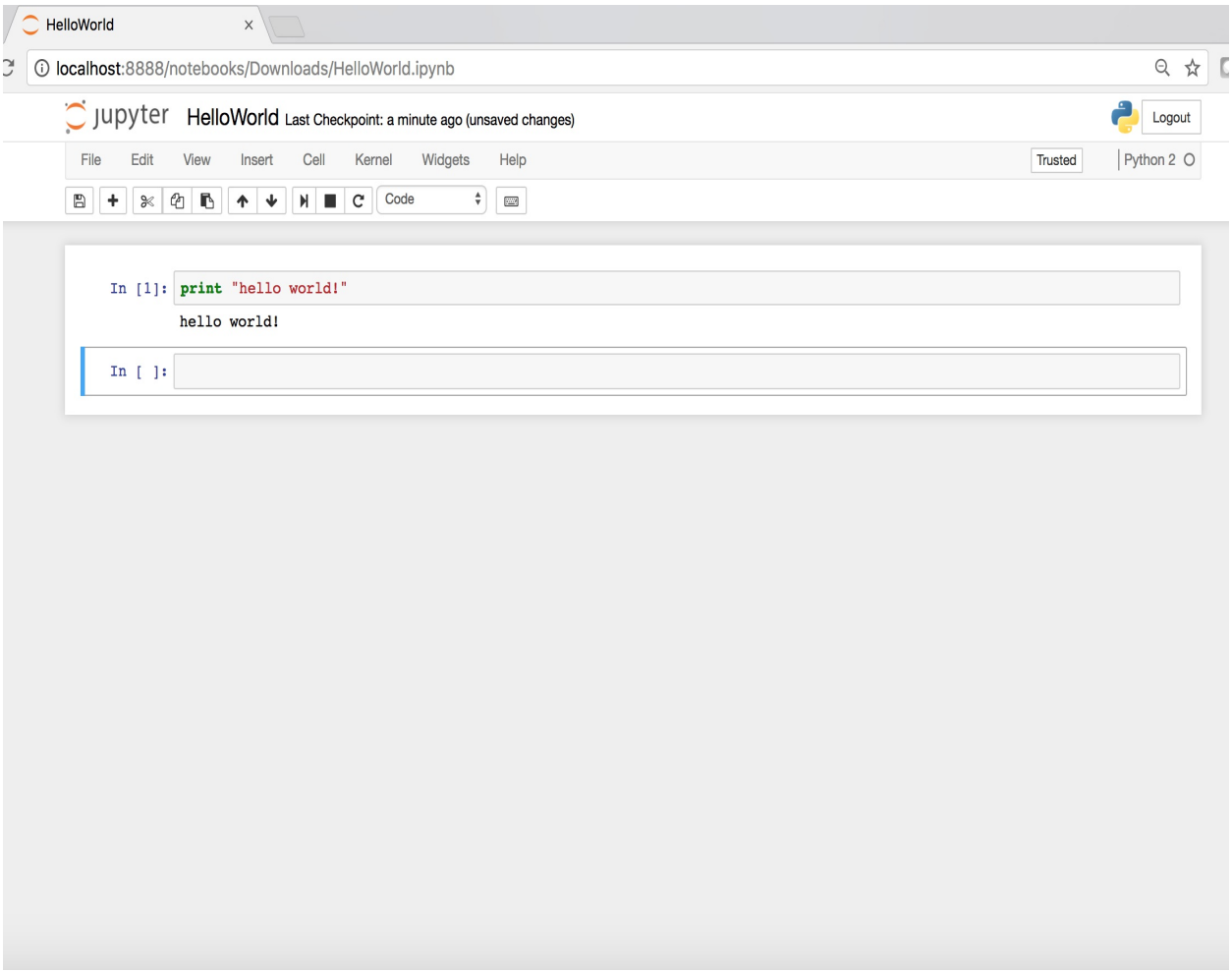
Jupyter Notebook can also be installed by running the following commands:

```
pip install --upgrade pip
```

```
pip3 install jupyter
```

Incase if the user is on Python 2 pip3 needs be replaced by pip

On installation you can just type "jupyter notebook" to run it. This opens the Jupyter Notebook in the primary browser of the system.Alternatively you can open Jupyter from the Anaconda Navigator.


HelloWorld
localhost:8888/notebooks/Downloads/HelloWorld.ipynb
jupyter
HelloWorld
Last Checkpoint: a minute ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 2
Code
In [1]:
print "hello world!"
hello world!
In [ ]:

Python Packages

In the section below we discuss packages that form the backbone for Python's machine learning architecture.

numPy

numpy is a free python package that is used to perform any computation task, Numpy is abosultely important when doing statistical analysis or machine learning. Numpy contains sophisticated functions for solving linear algebra, Fourier transform and other numerical analysis. Numpy can be installed by running the following:

```
pip install numpy
```

to install this through jupyter use the folowing:

```
import sys
!{sys.executable} -m pip install numpy
```

sciPy

Scipy is a python package that is created on top of the numpy array object. Scipy contains an array of functions like integration, linear algebra, image processing functionalities and so on. Like numpy scipy can also be installed likewise. numply and scipy are generally used together.

To check the version of scipy installed in your system you can run the following code:

```
import scipy as sp
print ("SciPy version:{}",format(sp.version))
```

scikit-learn

scikit learn is a free python package that is also written in python. scikit-learn provides a machine learning library that supports several popular machine learning alrotithms for classification, slustering, regression and so on. scikit-learn is very helpful for machine learning novices. scikit learn can be easilly installed by running the follwoing:

```
pip install sklearn
```

To check if the package is installed successfully test with the following piece of code in jupyter notebook or python command line.

```
import sklearn
```

if the above throws no errors then the package has been sucessfully been installed

scikit-learn requires two dependent package numpy and scipy to be installed . We will discuss the functionalities of these in the following sections. scikit-learn comes with a few inbuilt dataset like:

1. Iris data set
2. Breast cancer dataset
3. Diabetes dataset
4. The Boston house prices dataset and others

Other public data sets from libsvm and svmlight can also be loaded:

```
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
```

A sample script that uses scikit-learn to load data :

```
from sklearn.datasets import load_boston
boston=datasets.load_boston()
```

Pandas

The pandas open source package that provides easy to data structure and data frame. These are powerful for data analysis and is used in statistical learning. The pandas data frame allows different data types to be stored alongside each other much unlike the numpy array where same data type need to be stored together.

matplotlib

Matplotlib is a package used for plotting and graphing purposes. This helps create visualizations in 2D space. Matplotlib can be used from the Jupyter Notebook , from web application server or from the other user interfaces.

Let us plot a a small sample of the iris data that is available in the sklearn library. The data has 150 data samples and the dimensionality is 4.

We import the sklearn and matplotlib libraries in our python environment and check the data and the features

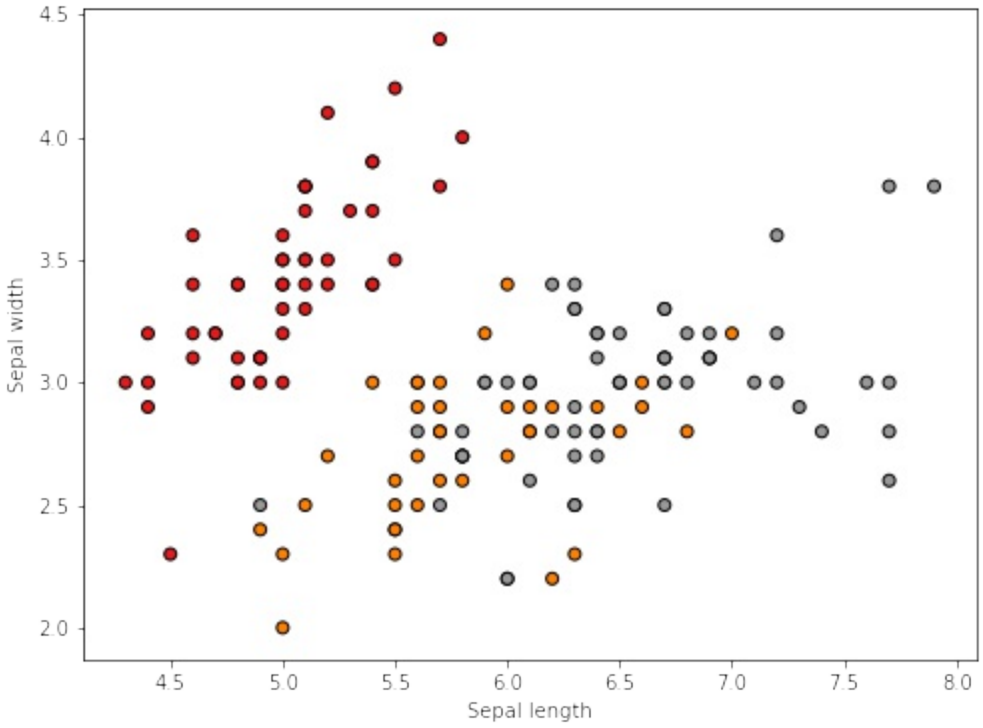
```
import matplotlib.pyplot as plt
from sklearn import datasets
iris = datasets.load_iris()
print(iris.data.shape) # gives the data size and dimensions
print(iris.feature_names)

Output:
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

We extract the first two dimensions and plot it on an X by Y plot as follows:

```
X = iris.data[:, :2] # plotting the first two dimensions
y = iris.target
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
plt.figure(2, figsize=(8, 6))
plt.clf()plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1,
 edgecolor='k')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
```

We get a plot like below:


4.5
4.0
3.5
3.0
2.5
2.0
Sepal width
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length

Mongodb with Python

MongoDB can store unstructured data that is fast and capable of retrieving large amount of data over a small time. MongoDB uses a json format to store data in rows. Thus any data without a common schema can be stored. We would be using MongoDB in the next few chapters because of its distributed nature. MongoDB has a fault tolerant distribution by shredding the data into multiple servers. MongoDB is generates primary key as you store data.

Installing MongoDB

To install MongoDB on your Windows, Mac or Linus systems run by the following steps:

- Download MongoDB from the download center in the following link for a windows or mac system:

```
https://www.mongodb.com/download-center
```

- On a linux system you can download it from:

```
sudo apt-get install -y mongodb-org
```

- MongoDB requires a separate repository of it own where you can extract the and store the contents upon installation
- Finally you can start the MongoDB service

PyMongo

To use MongoDB from within Python we will be using the PyMongo Library. PyMongo contains tools that helps to work with MongoDB. There are libraries that act as object data mapper for mongodb, however PyMongo is the recommended one.

To install PyMongo you can either run the following:

```
python -m pip install pymongo
```

Alternatively you can use the following :

```
import sys
!{sys.executable} -m pip install pymongo
```

Finally you can get started with using MongoDB by importing the pymongo library and then setting up a connection with MongoDB:

```
import pymongo
connection = pymongo.MongoClient()
```

On creating a successful connection with MongoDB, you can continue with different operations like listing the databases present and so on.

```
connection.database_names() #list databases in MongoDB
```

Each database in MongoDB contains data in containers called 'collections'. You can retrieve data from these collections to pursue with your desired operation.

```
selected_DB = connection["database_name"]
selected_DB.collection_names() # list all collections within the selected database
```

Setting up the development and testing environment

In this section we will discuss how setup a machine learning environment. This starts with a usecase that we are trying to solve, once we have short listed the problem we select the IDE where we will do the the end to end coding.

We need to procure a data set and divide the data into testing and training data. Finally we finish the setup of the environment by importing the ideal packages that are required for computation and visualization.

Since we deal with machine learning usecases for the rest of this book, we choose our usecase in a different sector. We will go with the most generic example i.e. prediction of stock prices. We use a standard dataset with xxx points and yy dimensions.

Usecase

We come up with a usecase that predicts the onset of given few features by creating a stock predictor that ingests in a bunch of parameters and uses this to make a prediction.

Data

We can use multiple data sources like audio, video or textual data to make such a prediction , however we stick to a single text data type. We use sckit-learn's default diabetes data set to to come up with a single machine learning model i.e. regression for doing the predictions and error analysis.

Code

We will use an open source code available from scikit-learn site and use it for this case study. The link to the code is available :

```
http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py
```

We import packages

- matplotlib
- numpy
- sklearn

Since we will be using regression for our analysis, we import the linear_model , mean_square_error and r2_score libraries.

```
print(__doc__)
# Code source: Jaques Grobler
# License: BSD 3 clause
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

We import the diabetes data and perform the following actions

- List the dimension and size
- List the features

The associated code for the above is:

```
# Load the diabetes dataset
diabetes = datasets.load_diabetes()
print(diabetes.data.shape) # gives the data size and dimensions
print(diabetes.feature_names
print(diabetes.DESCR)
```

The data has 442 rows of data and has 10 features. The features are:

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

To train the model we use a single feature i.e. the bmi of the individual

```
# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 3]
```

Earlier in the chapter we discussed that selecting a proper training and testing set are integral. The last 20 items is kept for testing in our case.

```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]#everything except the last twenty itemsdiabetes_X_test =
diabetes_X[-20:]#last twenty items in the array
```

Further we also split the targets into training and testing sets

```
# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]

everything except the last two items


diabetes_y_test = diabetes.target[-20:]
```

Next we perform regression on this data to generate results. We use the testing data to fit the model and then then use the testing data set to make prediction on the test data set that we have extracted.

```
# Create linear regression object
regr = linear_model.LinearRegression()
#Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

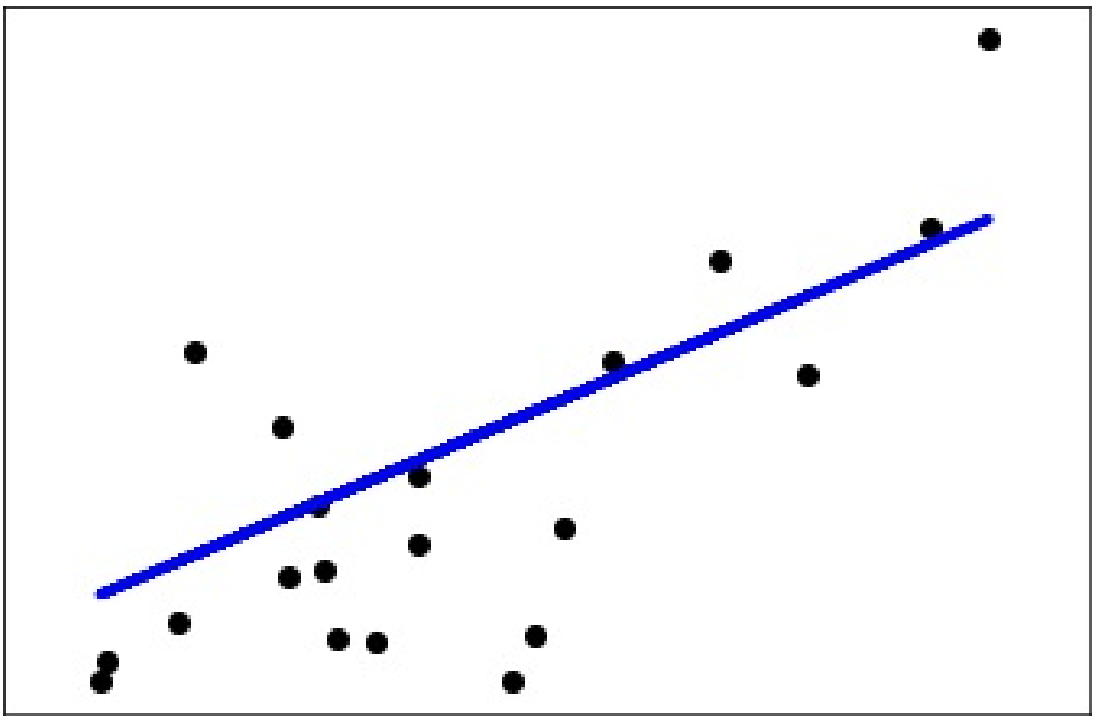
We compute the goodness of fit by computing how large large or small the errors are by computing the mean squared error and variance

```
# The mean squared error
print("Mean squared error: %.2f"
 % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
```

Finally we plot the prediction using the matplotlib graph as follows:

```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```

The output graph looks like :



Summary

In this chapter of the book we have gone the basics of machine learning. We briefly discussed how machine learning fits into daily use-cases and its relation with the cyber security world. The chapter discusses the different aspects of data that is required to be known to deal with machine learning. We discussed the different segregation of machine learning and the different machine learning algorithms. The chapter also discusses real world platforms that are available on this sector. Finally the chapter discusses the hands on aspects of machine, installation of IDE, installation of packages and setting up the environment for work. The chapter finally ends by taking up an example and working on it from end to end.

Time Series Analysis and Ensemble Learning

In this chapter we study two important concepts of machine learning, the time series analysis and the ensemble learning. These are important concepts in the field of machine learning

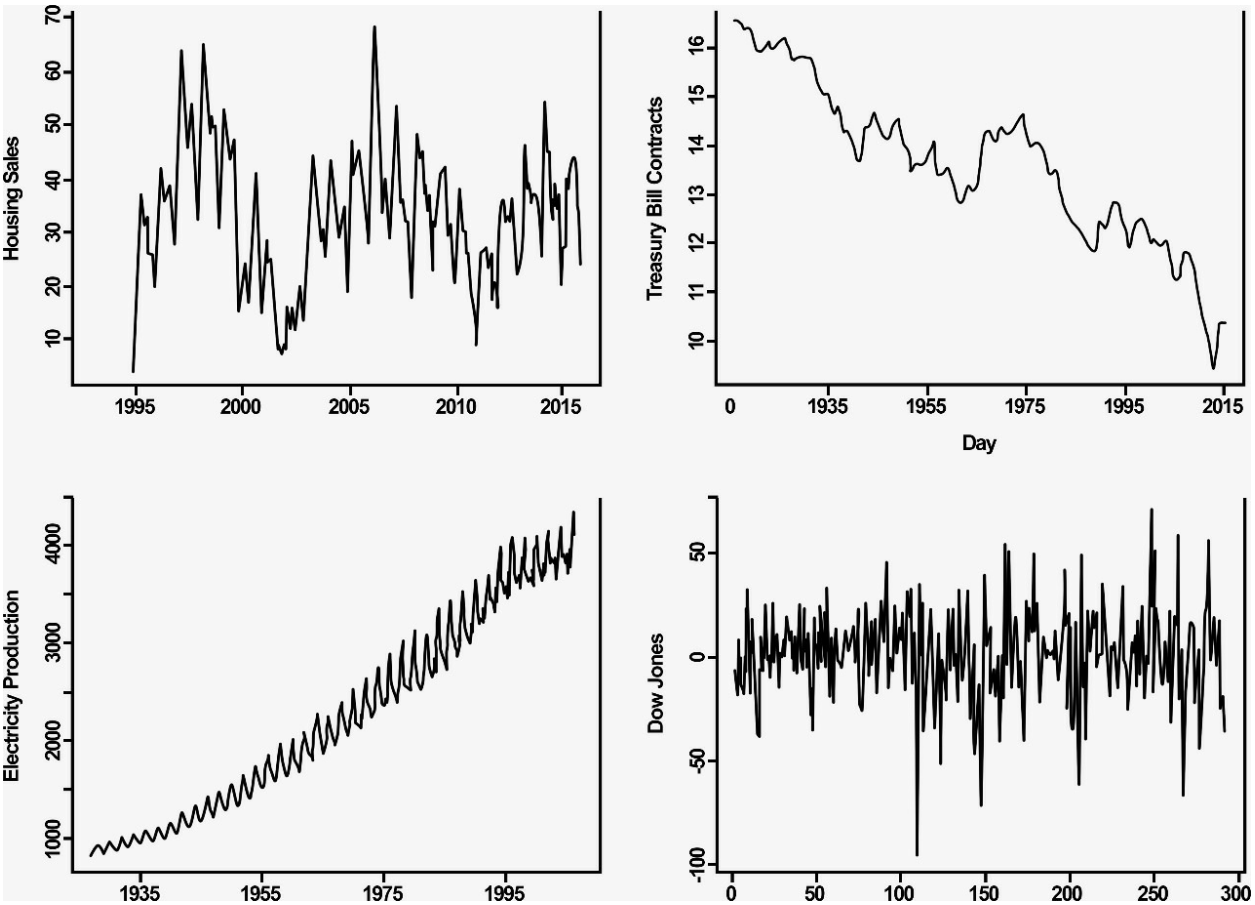
We use these concepts to detect anomalies within a system. We analyze the historic data and compare it with the current data to detect deviations from normal activities.

What is time series?

Time series is defined as an array of data points that are arranged in order with respect to time. The data points are indicative of an activity that take place at a time interval. One popular example is the total number of stocks that were traded at a certain time interval with other details like stock prices and their respective trading info at each second. Unlike continuous time variable these time series data points have a discrete value at different points of time. Hence these are often referred to as discrete data variables. Time series data can be gathered over any minimum or maximum amount of time. There is no upper or lower bound to the period over which data is collected.

A time series data has:

1. Specific instances of time forming the time stamp
2. A start time stand and an end time stamp
3. The total elapsed time for the instance

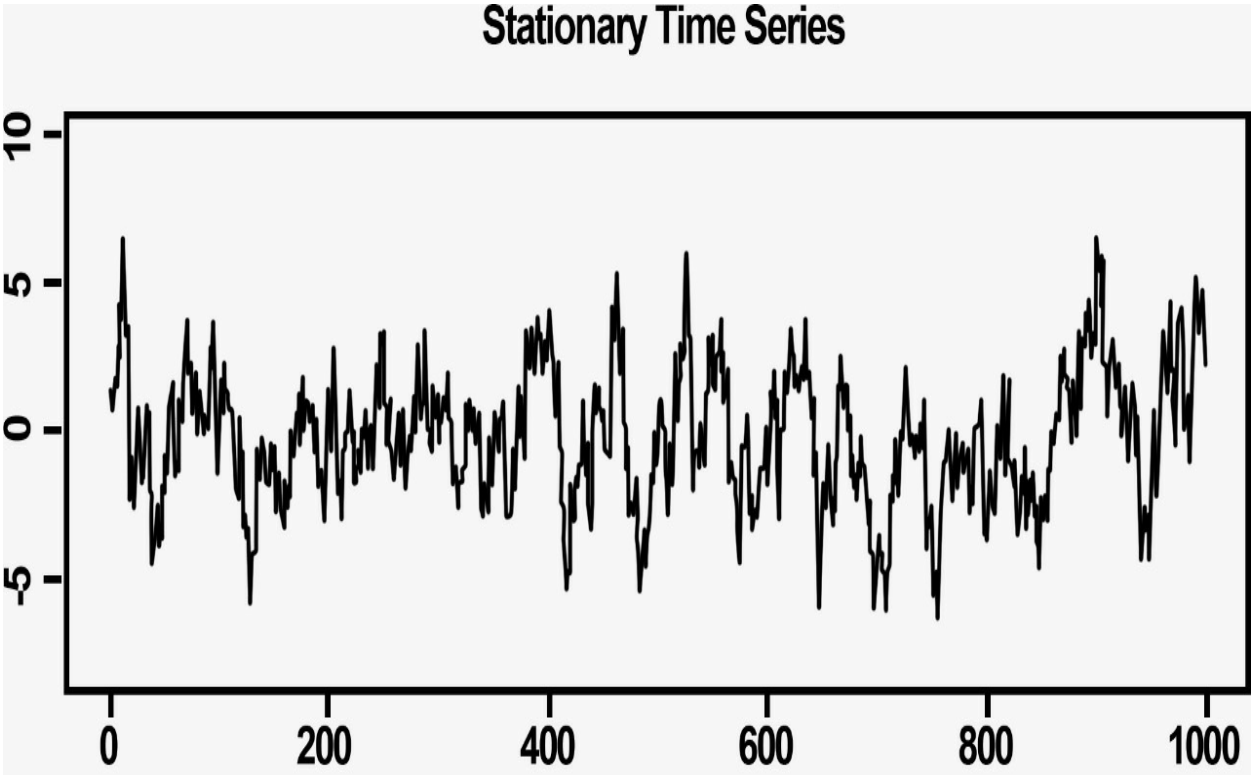

Housing Sales
10
20
30
40
50
60
70
1995
2000
2005
2010
2015
Treasury Bill Contracts
10
11
12
13
14
15
16
0
1935
1955
1975
1995
2015
Day
Electricity Production
1000
2000
3000
4000
1935
1955
1975
1995
Dow Jones
-100
-50
0
50
0
50
100
150
200
250
300

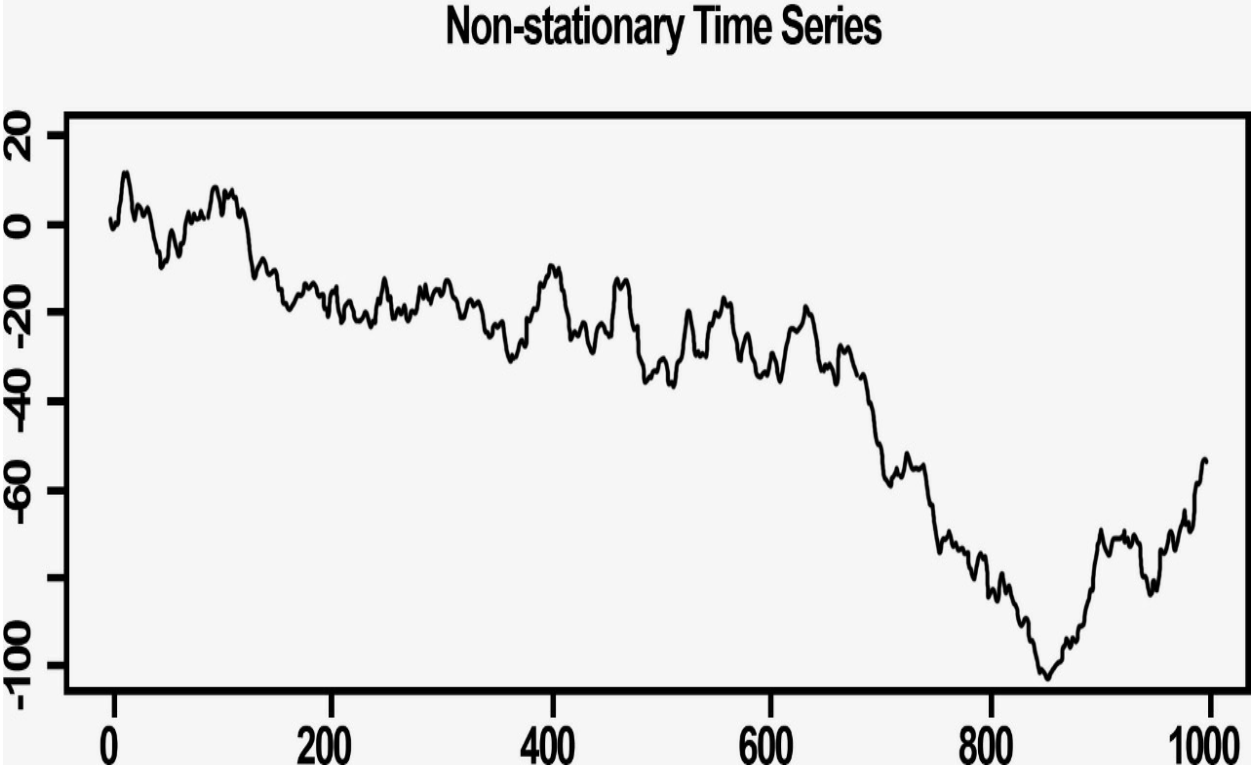
What is time series analysis ?

Time series analysis is the study where time series data points are mined and investigated. It is a method of interpreting quantitative data and computing the changes that it has undergone with respect to time. Time series analysis involves both univariate and multivariate time analysis. These sorts of time based analysis are used in many areas like signal processing, stock market predictions, weather forecasting, demographic related predictions and cyber attacking detections.

Stationarity of a time series models

A time series needs to be stationary or else building a time series model on top is not technically possible . This can be called a prerequisite for model building. For a stationary time series the mean , variance and auto correlation are consistently distributed over time.


Stationary Time Series
10
5
0
-5
0
200
400
600
800
1000


Non-stationary Time Series
20
0
-20
-40
-60
-100
0
200
400
600
800
1000

Strictly stationary process

A time series T is called strongly or strictly stationary if two or more random vectors have equal joint distribution for all indices and integers.

Random Vector 1 = { Xt1, ..., Xtn}

Random Vector 2 = { Xt1 +s , ..., Xtn +s}

s = all integers

t1..tn = all indices

Weak or Wide Sense Stationarity

A time series T is called weakly stationary if it has a shift invariance for the first and second moments of the process

Correlation in Time Series

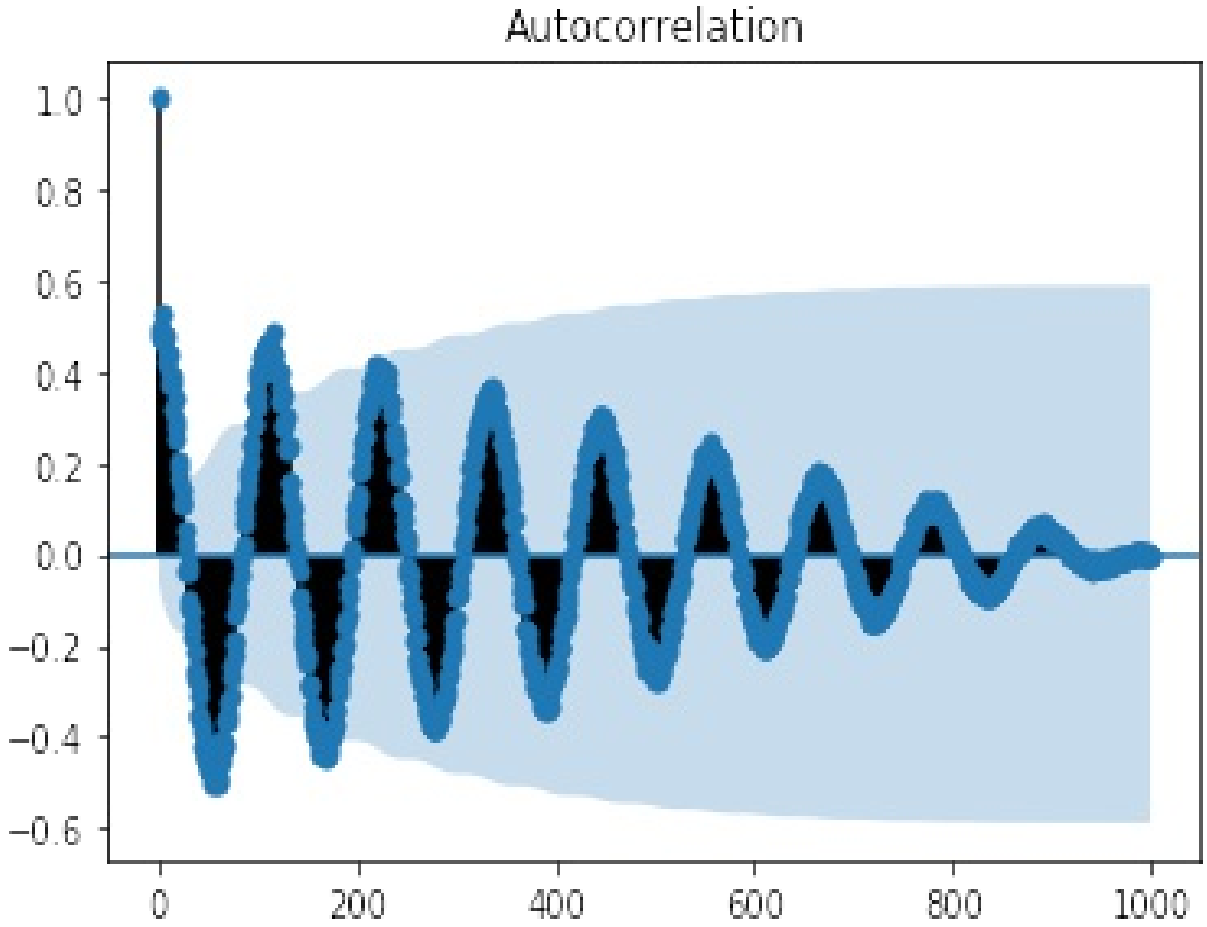
Autocorrelation

In order to choose two variables as a candidate of time series modeling, we are require to perform a statisticl correlation analysis between the said variables. Here each variable fir the Gaussian Curve and the Pearson's Coefficient is used to identify the correlation that exists between two variables.

In time series analysis the autocorrelation is measures with respect o historic data called lags. An auto correlation function or ACF is used to plot such correlations with respect to lag.

In Python the Auto Correlation function is computed as follows:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as p
from statsmodels.graphics.tsaplots import plot_acf
data = p.Series(0.7 * np.random.rand(1000) + 0.3 * np.sin(np.linspace(-9 * np.pi, 9 * np.pi, num=1000)))
plot_acf(data)
pyplot.show()
```



Partial auto correlation function

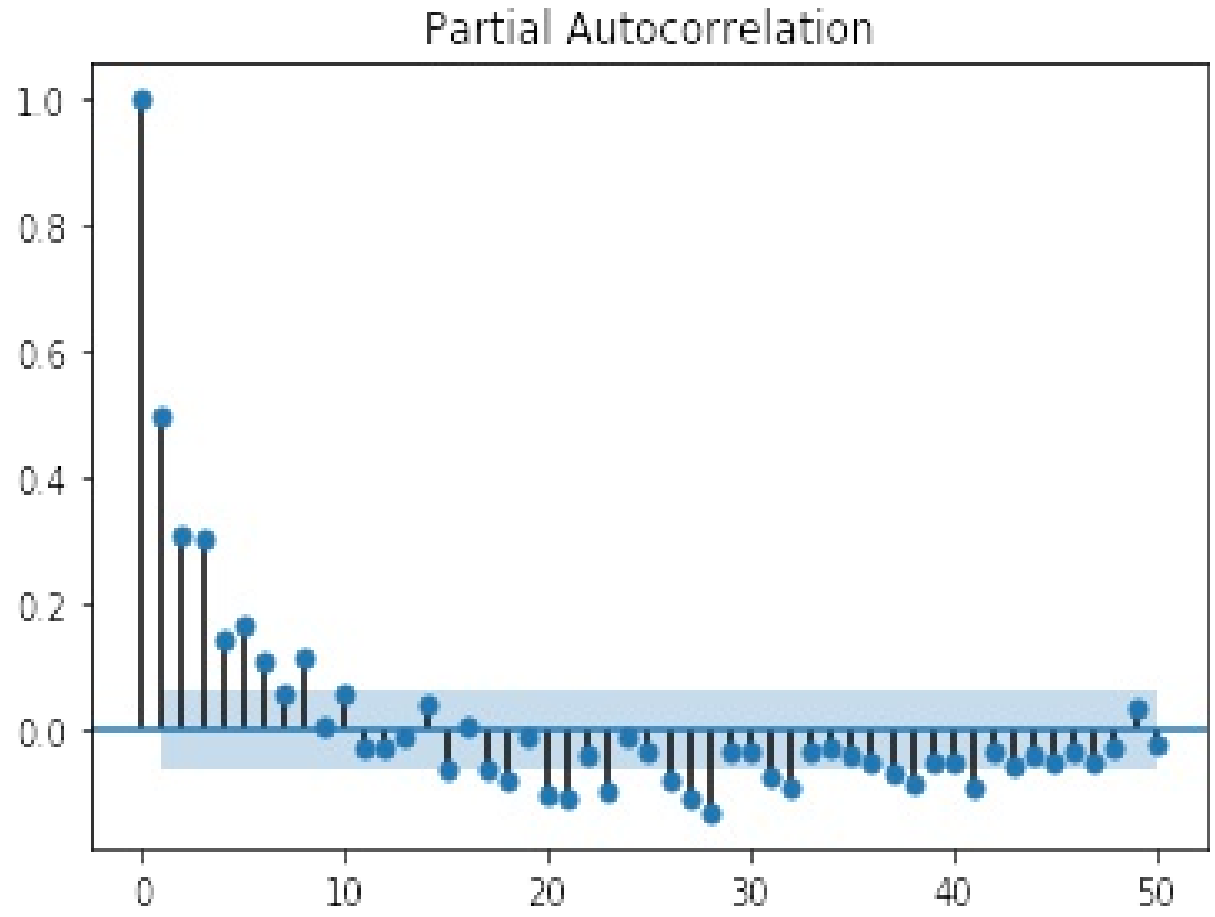
Let us start with wikipedia definition of the partial auto correlation function.

""In time series analysis, the partial autocorrelation function (PACF) gives the partial correlation of a time series with its own lagged values, controlling for the values of the time series at all shorter lags""

PACF is very much unlike ACF , here the auto correlation of a data point at the current point and auto correlation at a period lag have a direct or indirect correlation. PACF concepts are heavily used in auto regressive models.

In python the PACF function can be computed as follows:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as p
from statsmodels.graphics.tsaplots import plot_pacf
data = p.Series(0.7 * np.random.rand(1000) + 0.3 * np.sin(np.linspace(-9 * np.pi, 9 * np.pi, num=1000)))
plot_pacf(data, lag = 50)
pyplot.show()
```



Classes of time series models

Following are the classes of time series modes:

Stochastic time series model

Stochastic processes are random mathematical objects that can be defined using random variables. These data points are known to be randomly changing over time. Stochastic processes can again be divided into threee main classes that are dependent on historic data points. They are Auto Regressive(AR) models, the Moving Average(MA) and integrated (I) models. These models combine to form the Auto Regressive Moving Average(ARMA), Auto Regressive Integrated Moving Average and Auto Regressive Fractional Integrated Moving Average. We will use these in the later sections of the chapter

Artificial Neural Network time series model.

Artificial Neural Network(ANN) are an alternative to stochastic processes in time series models. ANN helps in forecasting by regularity detection and pattern recognition. It uses this intelligence to detect seasonalities and helps in generalization of the data. Unlike Stochastic models like multi layer perceptrons , feed forward neural network (FNN) and time lagged neural network(TLNN) are mainly used in non linear time series model.

Support vector time series models

Support Vector Machine(SVM) is another accurate non-linear technique that can be used to derive meaningful insight from time series data. They work best when the data in non-linear and non-stationary. Unlike other time series models SVMs can predict without requiring historic data.

Time series components

Time Series model can be divided into components based on the level of abstraction. These components are the systematic components and the non-sytematic components.

Systemetic Models

These are time series models which have recurring properties and the data points show consistency. Hence they can be easily modeled.

Non-systemetic model

These are time series model that lack the presence of seasonal properties and thus cannot be easily modeled.

Time series decomposition

Time Series Decomposition is a better way of understanding the data in hand. Decomposing the model creates an abstract model that can be used for generalization of the data. Decomposition stands for identifying the trend, seasonal, cyclical and irregular components of the data. Making sense of data with these components is the systematic type of modeling.

In the section below we will look at these recurring properties and how they help analyse time series data.

Level

We discussed moving averages with respect to time series before. Level can be defined as the average or mean of a bunch of time series data points.

Trend

Values of data points in a time series either keep decreasing or increasing with time. They may also follow a cyclic patterns. Such increase or decreases in data point values are known as the trend of the data

Seasonality

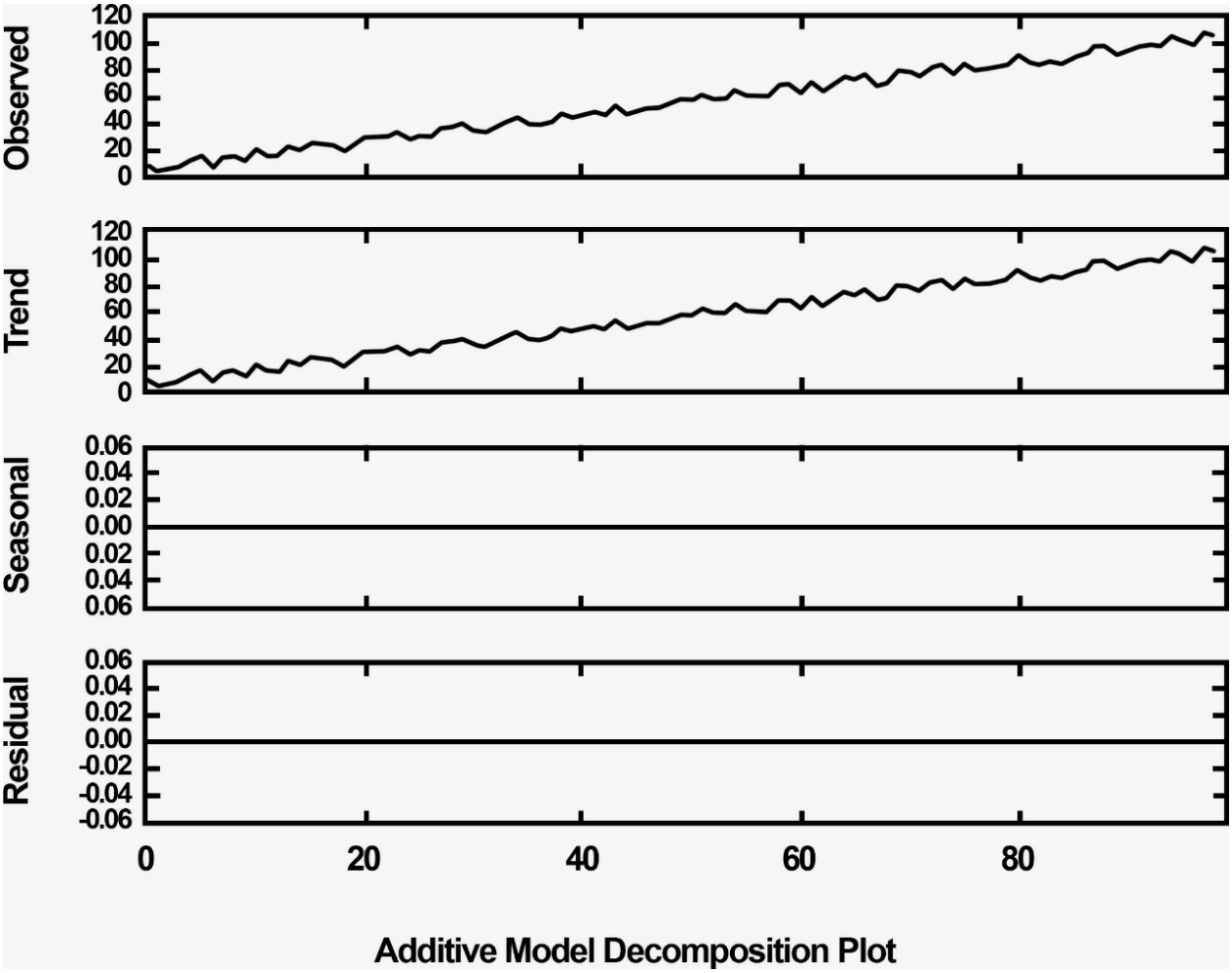
The values of data points also an increase or decrease that are more periodic such pattern are called seasonality. An example of this behavior could be a toy store , where there is increase and decrease in the amount of toys sold but in the thanks giving season that is November every year there is a spike in the sale that is unlike the increase or decrease seen rest of the year.

Noise

These are random increase or decrease of values in the series.

We will be generally dealing with the above systematic components in the form of addictive models, where additive models can be defined as the sum of level, trend, seasonality and noise. The other type is called the multiplicative model where the components are products of each other.

The graphs below help to distinguish between additive and Multiplicative Models.

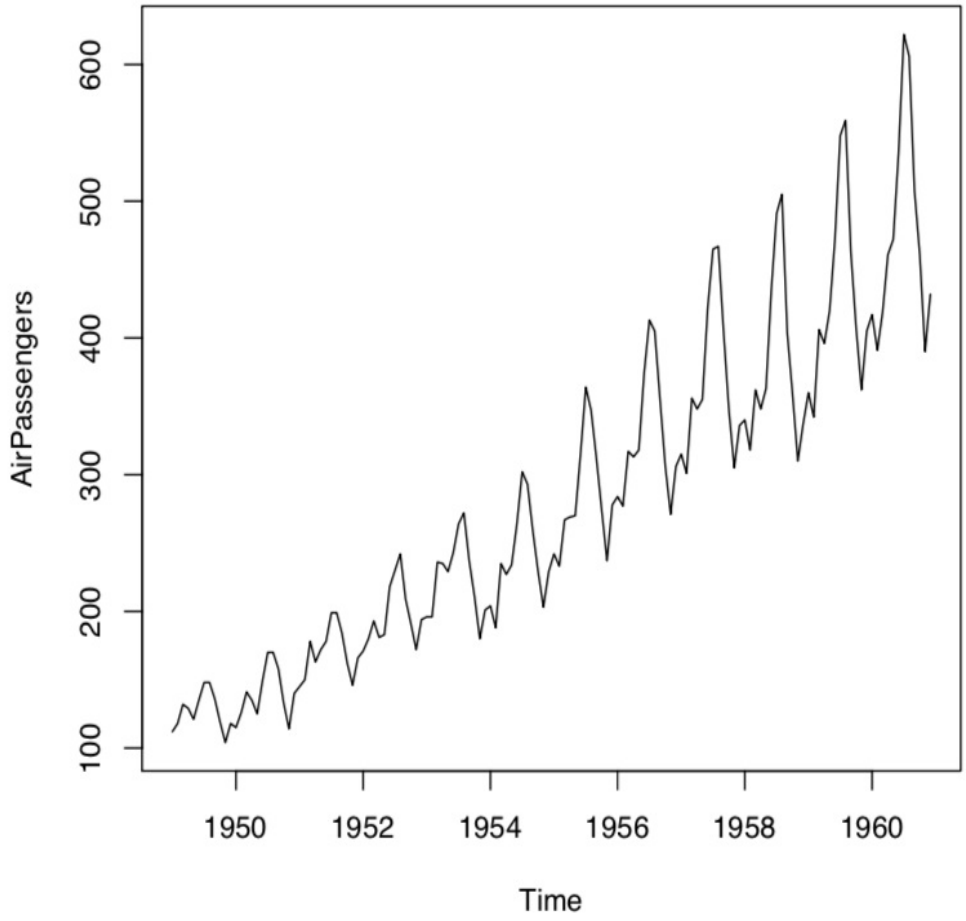


Since data decomposition has a major role in data analyzing thus we will understand these different components by using Pandas inbuilt data set that is the 'International airline passengers: monthly totals in thousands. Jan 49 – Dec 60' data set. The data set contains a total of 144 observation of sales from the period of 1949 to 1960 for Box & Jenkins

Lets us import and plot the data:

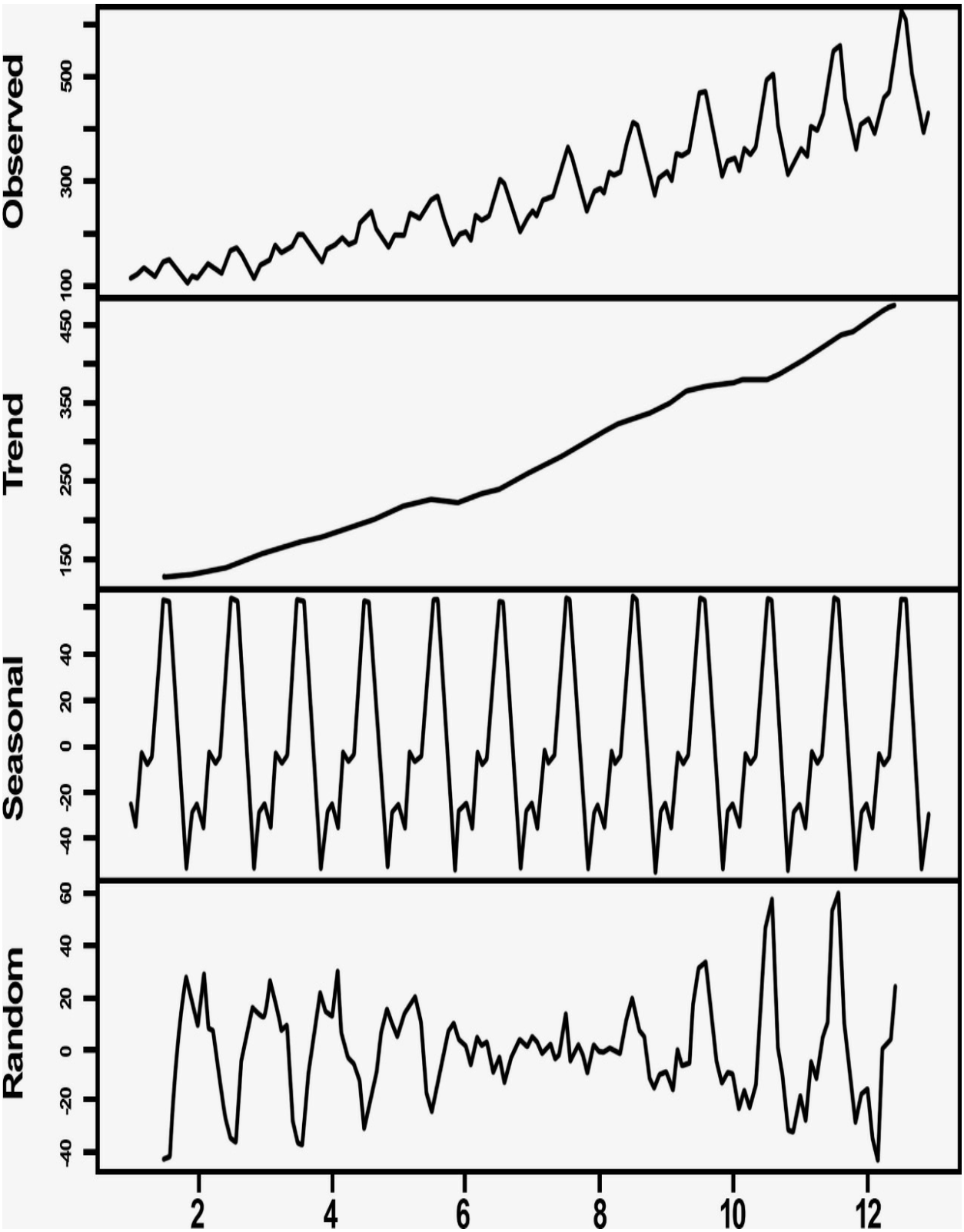
```
from pandas import Series
from matplotlib import pyplot
airline = Series.from_csv('/path/to/file/airline_data.csv', header=0)
airline.plot()
pyplot.show()
```

The graph below show how their is seasonality in data and there is a subsequent in crease in the height(amplitude) of the graph as years have progressed.



We can mathematically compute the trend and the seasonality for the above with an additive model.

```
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose
airline = Series.from_csv('/path/to/file/airline_data.csv', header=0)
result = seasonal_decompose(airline, model='additive')
result.plot()
pyplot.show()
```

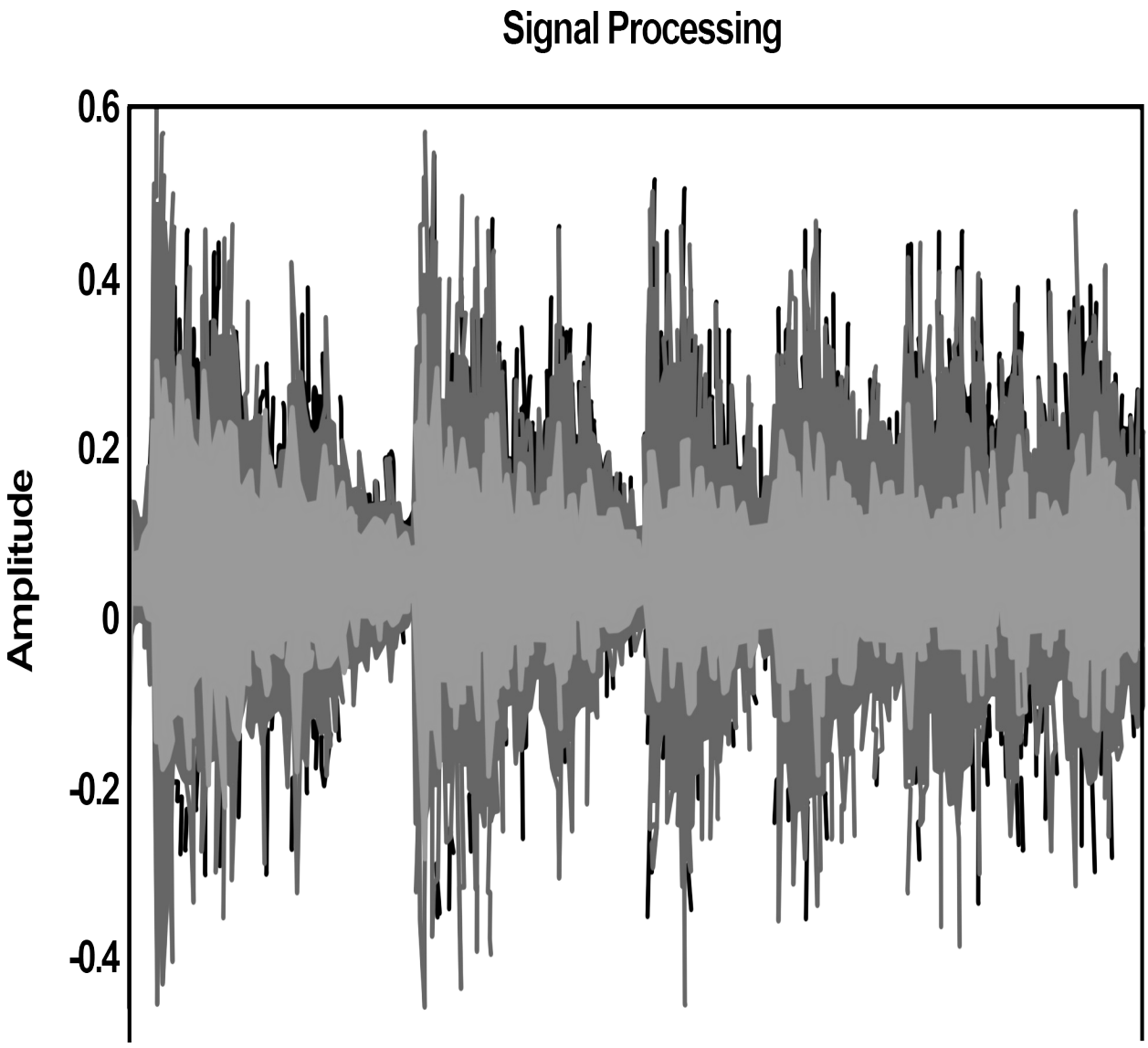


The above graph can be interpreted as follows:

- Observed - The Regular Arline Data Graph
- Trend - The observed increase in Trend
- Seasonality - The observed seasonality in the data
- Random - This is first observed graph after removing the initial seasonal patterns

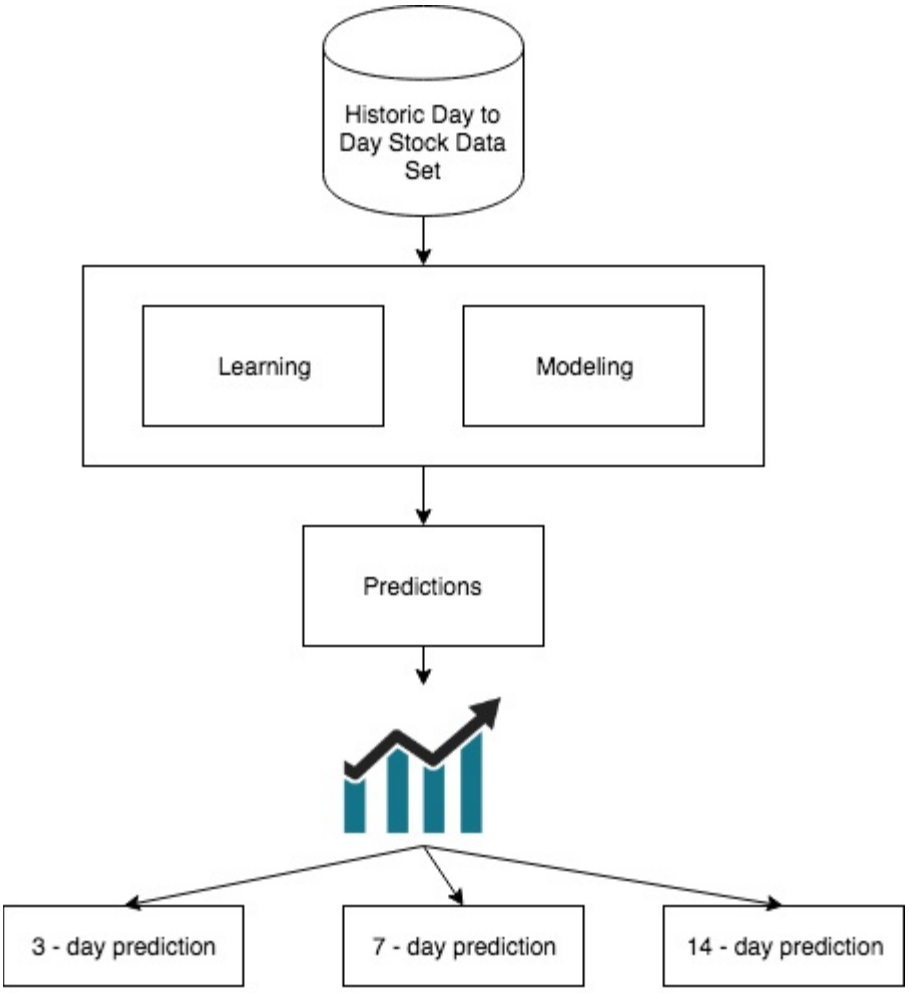
Usecases of time series

Digital signal processing uses time series analysis to effective identify a signal from mixture of noise and signals. Signal processing uses various methods to perform this identification like smoothing, correlation, convolution and so on. Time series helps measuring deviations from the stationary behaviors of signals. These drifts or deviations are the noises.



Stock market predictions

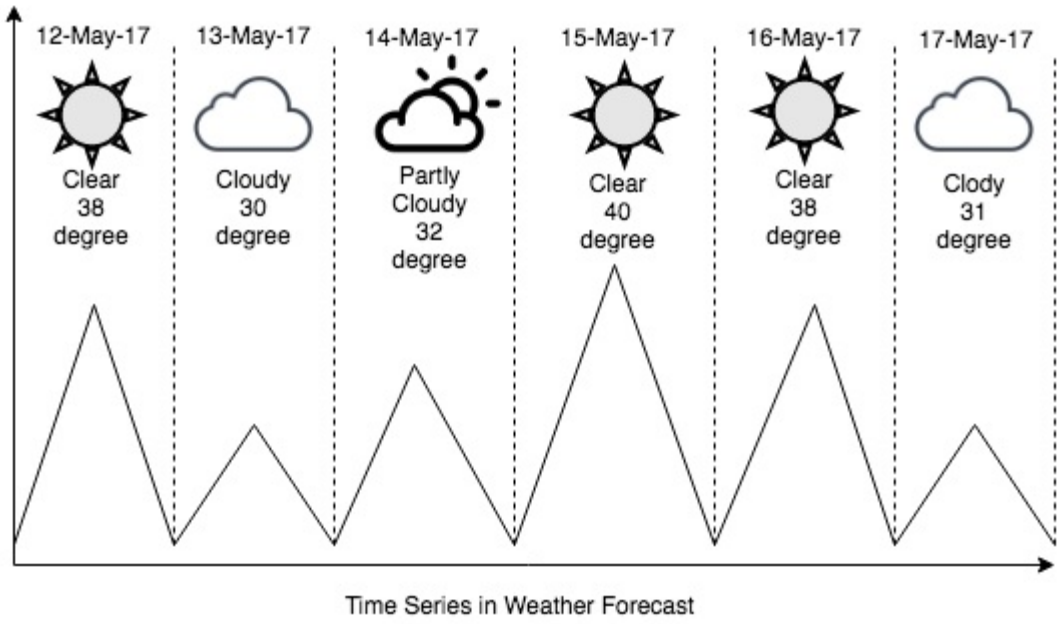
Stock market predictions are yet another use-case of time series analysis. Investors can make educated guess for stocks by analyzing data. Though non mathematical components like the history of the company do have a hand in stock market predictions but they largely depend on the historic stock market trends. We can do a trend analysis of historic trade-in prices of different stocks and use predictive analytics to predict future prices. Such analytics needs data points that is stock prices at each hour over a trading period. Quantitative Analysts and trading algorithms make investment decisions using these data points by performing time series analysis.



Time Series in Predicting Historic Stock Data

Weather forecasting

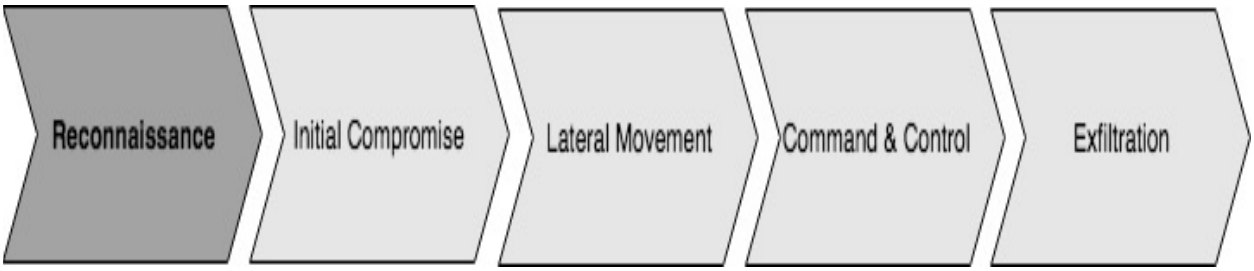
Time series analysis is majorly used process in the field of meteorology. The temperature data points obtained over a period of time helps to detect the nature of possible climate changes. They consider the seasonality in the data to predict weather changes like storms, hurricanes or blizzards and then adapt precautions to mitigate from the harmful side effects.



Time Series in Weather Forecast

Reconnoissance detection

We can use time series concepts to detect early signs of malware compromise or cyber attack against a system. In the earliest phase of attack the malware just sniffs through the system looking for vulnerabilities. The malware goes looking for loosely open ports and peripherals, thus snooping information about the system. These early stages of cyber attack are very similar to what the military does when it surveys a new region looking for enemy activities. This stage in cyber attack is called the reconnaissance.



Time series analysis in cybersecurity

Computer attacks interrupt day to day services, causes data losses and network interruption. Time series analysis are popular machine learning methods that helps to quantitatively detect anomalies or outlier in data by either data fitting or forecasting. Time series analysis helps in thwarting compromise and keeping information loss to a minimum.

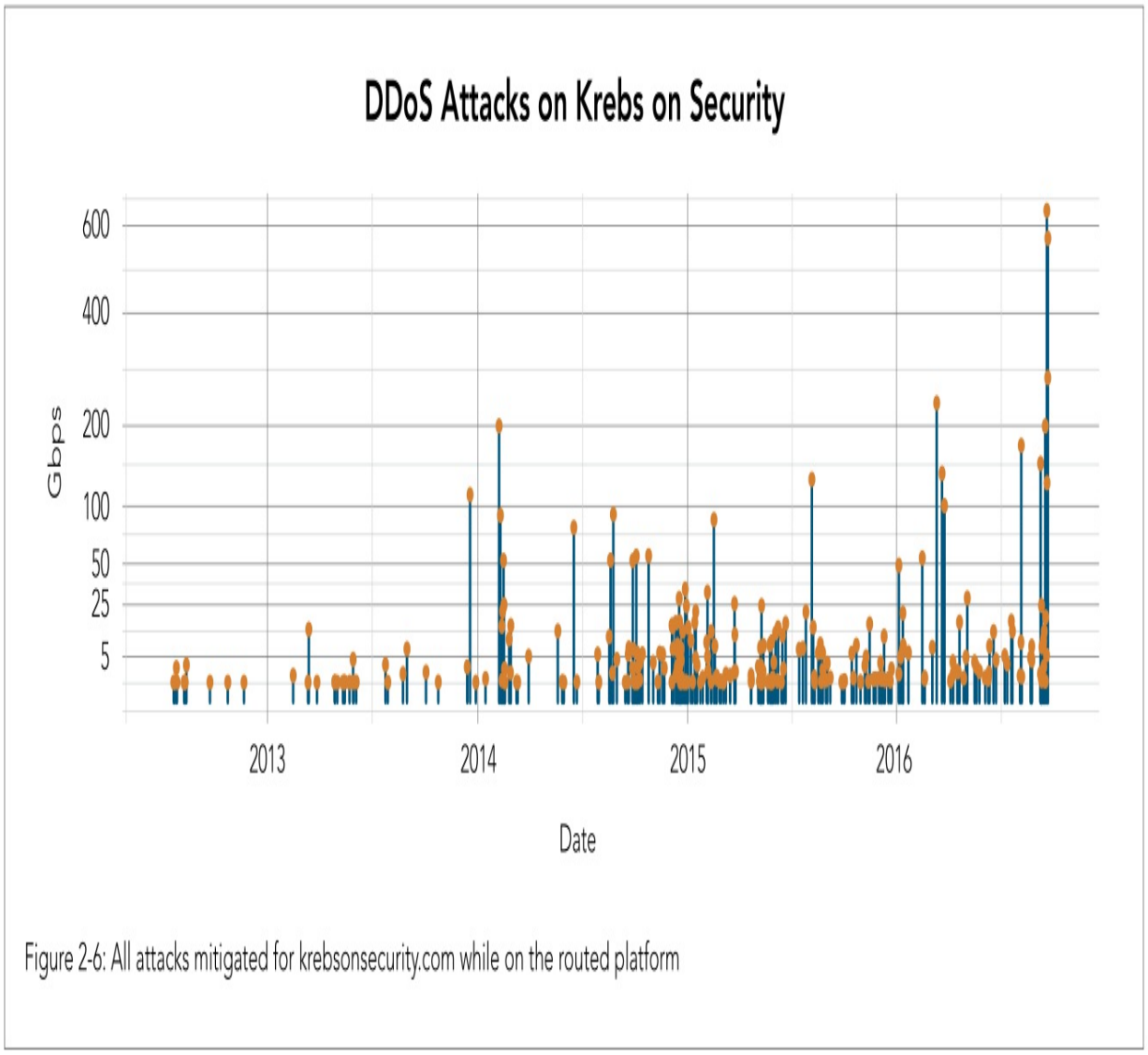


Figure 2-6: All attacks mitigated for krebsonsecurity.com while on the routed platform

Time series analysis can be used to detect attack attempts like failed login, using a time series model. Plotting login attempts to identify the spikes in failed login /. Such spikes are indicatives of account take over(ATO).

Time series is identifying another cyber security usecase- Data Exfilteratiion. Data exfilteration

is the process in which unauthorized tranfer of data from computer system. Time series can idemtify hige network data packets being transported out of the network. Data exfiltertaion could be either because of a outsider compromise or an insider threat. In the later sectionof the chapter we would use emsemble learning methods to identify the source of the attack.

We will learn the details of the attack in the next section. The goal of this chapter is to be able to detect reconnaissance so that we are able to thwart compromise to the early stages and keep the loss of information to a minimum.

Detecting distributed denial of series with time series

Distributed Denial of service is a cyber security menace which disrupts online services by sending overwhelming amount of network traffic. These attacks are manually started with botnets that flood the target network. these attacks could either have the following characteristics.

- The Botnet Sends massive number of requests to the hosting servers
- Botnet sends high volume of random data packets thus incapacitating the network

Time series analysis helps identifying network pattern with respect to time. Such pattern detection is done with historic monitoring of network traffic data. It helps identifying attacks like Distributed Denial of service (DDOS). These attacks can be very critical if done. Baselining the regular traffic of a network and then overlaying the network a compromised activity on top of it will help detecting the deviation from the normal.

We will be analyzing this usecase and choose a machine learning model that will help detect such DDOS attacks before they crash the entire network.

We will work with a data set that compromises of traffic received by a website say "donotddos.com". We will analyze thirty days of historic network traffic data from this website and detect whether the current traffic being recieved by the website is a part of any DDOS attack or not.

Before we go into the details of this usecase we will analyze the "datatime" data type of Python since it will form the building block of any time series model.

Dealing with the Time element of time series

Import the inbuilt python package 'datetime'

```
from datetime import datetime
```

To get the current date time that is the timestamp do the following:

```
timestamp_now = datetime.now()
```

```
Output: datetime.datetime(2018, 3, 14, 0, 10, 2, 17534)
```

You can get the difference between two timestamps by just doing the following:

```
time_difference = datetime(2018,3,14) - datetime(2015,2,13,0,59)
```

```
Output: datetime.timedelta(1124, 82860)
```

You can extract the day from the above by :

```
time_difference.days = 1124
```

You can extract the seconds by :

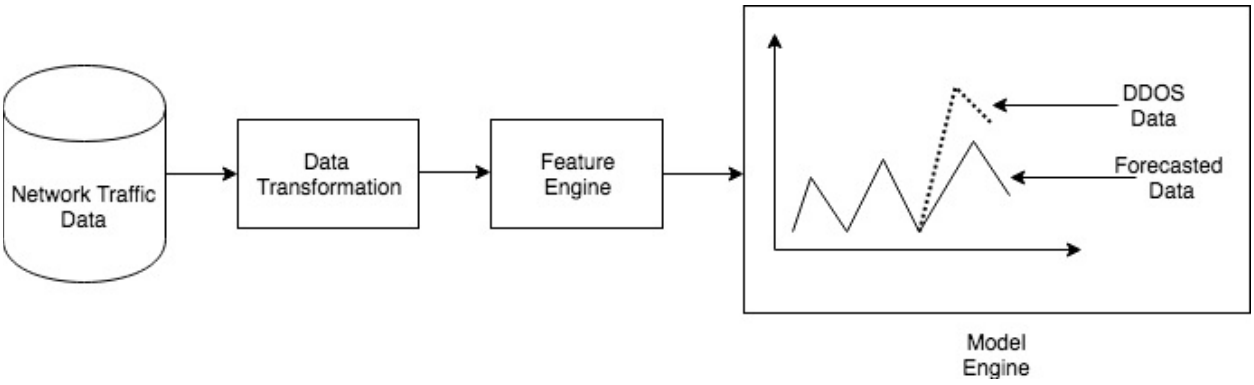
```
time_difference.seconds = 82860
```

Datetime can also be added to each other similarly like the subtraction process shown above.

Tackling the Usecase

The use-case undergoes through the following stages:

1. We start with importing our data in a pandas data frame
2. We determine that the data is properly cleansed
3. We analyse the data, as per the model requirement
4. We extract features from the data and analyze the features again to measure the correlation, variance, seasonality etcetera.
5. We will then fit a time series model to predict whether the current data is a part of an DDOS attack or not. Fig belows sums up the entire procedure.



Importing packages

We import the relevant python packages that will be need for the visualization of this usecase.

```
import pandas as p
import seaborn as sb
import numpy as n
%matplotlib inline
import matplotlib.pyplot as pl
```

Importing data in Pandas

We have data in a csv file that is text file separated by comma. While importing the data we also identify the headers in the data. Since we deal with packet capture data from the network, the column captured are as follows:

1. Sl Num : Serial Number
2. Time : Time of Record Capture
3. Source : Source Address or origin of the network packet
4. Destination : Destination Address of the network
5. Volume : Data volume exchanged in Kilobyte(KB)
6. Protocol : The network protocol viz. SMTP, FTP, HTTP

```
pdata_frame = pd.read_csv("path/to/file.csv", sep=',', index_col = 'Sl Num', names = ["Sl Num", "Time", "Source", "Destination", "Protocol"])
```

Let us dump the first few lines of the data frame and have a look at the data. The code below displays the first 10 lines of the packet capture data set.

```
pdata_frame.head(n=9)
```

The output of the above is as follows:

Sl Num	Time	Source	Destination	Volume	Protocol
1	1521039662	192.168.0.1	igmp.mcast.net	5	IGMP
2	1521039663	192.168.0.2	239.255.255.250	1	IGMP
3	1521039666	192.168.0.2	192.168.10.1	2	UDP
4	1521039669	192.168.10.2	192.168.0.8	20	DNS
5	1521039671	192.168.10.2	192.168.0.8	1	TCP
6	1521039673	192.168.0.1	192.168.0.2	1	TCP
7	1521039674	192.168.0.2	192.168.0.1	1	TCP
8	1521039675	192.168.0.1	192.168.0.2	5	DNS
9	1521039676	192.168.0.2	192.168.10.8	2	DNS

Data cleansing and transaformation

Our Data set is largely clean so we will directly transform the data into more meaning-full forms. For example the timestamp of the data is in epoch format. Epoch format is alternatively known as Unix or Posix time format. We will convert this to the date time format that we have previously discussed.

```
import time
time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(1521388078))

Out: '2018-03-18 21:17:58'
```

We perform the above operation on the 'Time' column and add to a new column and call it 'Newtime'

```
pdata_frame['Newtime'] = pdata_frame['Time'].apply(lambda x: time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(float(x))))
```

Once we have transformed the data to a more readable format , we look at the other data columns. Since the other columns look pretty cleansed and transformed, we will leave them as is. The volume column is the next data that we will look into. We aggregate volume in the same by the hour and plot them with the code below:

```
import matplotlib.pyplot as plt
plt.scatter(pdata_frame['time'],pdata_frame['volume'])
plt.show() # Depending on whether you use IPython or interactive mode, etc.
```

Data analysis

To carry any further analysis on the data by aggregating features.

Here are the following features that we extract:

1. For any source , we compute the volume of packets exchanged per minute.
2. For any source , we count the total number of connection received per minute.

Feature Computation

Since our computations are done per minute, we round off the time to the nearest minute.

```
_time = pdata_frame['Time'] #Time column of the data frame
edited_time = []
for row in pdata_frame.rows:
            arr = _time.split(':')
            time_till_mins = str(arr[0]) + str(arr[1])
            edited_time.append(time_till_mins) # the rounded off time
source = pdata_frame['Source'] # source address
```

The output of the above is time rounded of to the nearest minutes that is 2018-03-18 21:17:58 will become 2018-03-18 21:17:00

```
Out: '2018-03-18 21:17:00'
     '2018-03-18 21:18:00'
     '2018-03-18 21:19:00'
     '2018-03-18 21:20:00'
     '2018-03-19 21:17:00'
```

We will count the number of connections established per minute for a particular source by iterating through the time array for a given source.

```
connection_count = {} # dictionary that stores count of connections per minute
for s in source:
    for x in edited_time :
        if  x in connection_count :
            value = connection_count[x]
            value = value + 1
            connection_count[x] = value
        else:
            connection_count[x] = 1
new_count_df #count # date #source
```

The dictionary connection_count gives the number of connections. The output of the above looks like below.

Time	Source	Number of Connections
2018-03-18 21:17:00	192.168.0.2	5
2018-03-18 21:18:00	192.168.0.2	1
2018-03-18 21:19:00	192.168.0.2	10
2018-03-18 21:17:00	192.168.0.3	2
2018-03-18 21:20:00	192.168.0.2	3

2018-03-19 22:17:00 192.168.0.2 3

2018-03-19 22:19:00 192.168.0.2 1

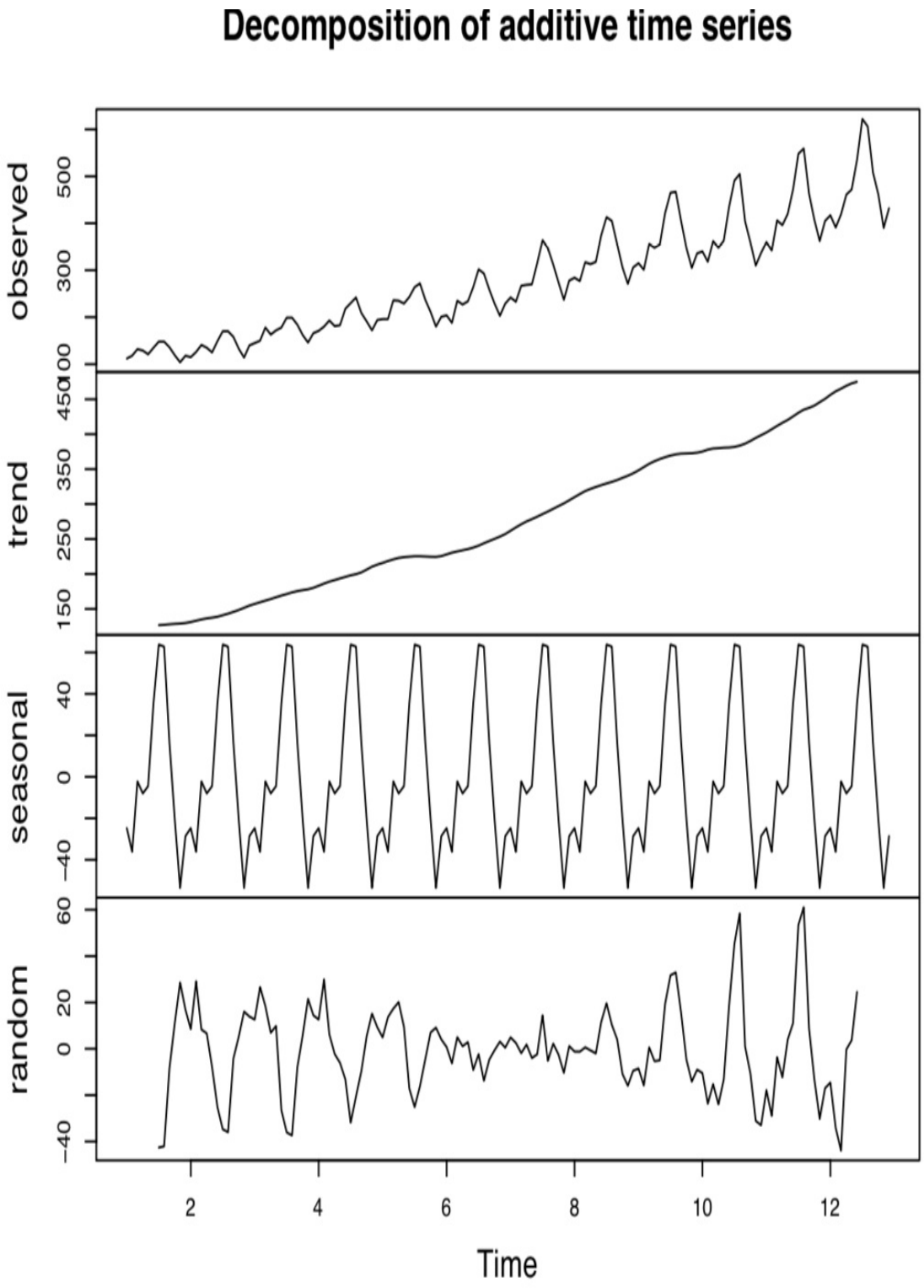
2018-03-19 22:22:00 192.168.0.2 1

2018-03-19 21:17:00 192.168.0.3 20

We will decompose the data with the following code to look for trends and seasonality in the data. Decomposition of the data helps in better detecting of an anomalous behavior viz. DDOS attack.

```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(new_count_df, model='additive')
result.plot()
pyplot.show()
```

The data generates a graph as follows, we are able to recognize the seasonality and trend of the data in general.



Next we find the ACF function for the data to understand the auto - correlation among the variables if any with the following piece of code:

```
from matplotlib import pyplot
```

```
from pandas.tools.plotting import autocorrelation_plot
autocorrelation_plot(new_count_df)
pyplot.show()
```

Predicting DDOS attack

Now that we have identified a seasonality and the trend in the network data we will baseline the data by fitting this to a stochastic model. We have earlier mentioned some of these, we will discuss these in greater details in the section bellow:

Autoregressive–moving-average(ARMA)

This is a weakly stochastic stationary process such that when provided with a time series, Xt, ARMA helps to forecast future with respect current values. ARMA consists of two actions.

1. The Autoregression (p)
2. The moving average (q)

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}.$$

C = Constant

Et = White Noise

theta = parameters

Autoregressive–integrated-moving-average(ARMA)

ARIMA are generalized version of ARMA. Thet either help undetstand the data or make predictions. These model can be applied to non stationary sets and hence require an initial differential step. ARIMA can be either seasonal pr non seasonal . ARIMA can be defined with (p,d,q) where :

p= order of the AR model

d = degree of the deferencing

q = order of the moving average

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t.$$

Where, Xt = The given time series

L = lag operator

Et = error terms

parameter for AR (wiki)

parameter for MA(wiki)

drift is $\delta/(1 - \Sigma\phi_i)$.

Autoregressive–fractional integrated-moving-average(ARMA)

These are generalized ARIMA model that allow non-integer values of differencing parameter. These are used in time series model with long memory.

Now that we have discussed the details of each of these stochastic models we will fit the model with the baseline network data. We will use the statsmodels library to use the 'ARIMA' stochastic model. We will pass the p,d,q values for the ARIMA model. The lag value for auto regression is set to 10, the difference in order is set to 1 and the moving average is set to 0. We use the 'fit' function to fit the training/baseline data.

```
# fitting the model
model = ARIMA(new_count_df, order=(10,1,0))
fitted_model = model.fit(disp=0)
print(fitted_model.summary())
```

The above is model that is fit in the entire training set and we can use if to find the residuals in the data. This method in a way helps us to understand the data better but is not used in forecasts. To find the residuals in the data we can do the following in python:

```
# plot residual errors
residuals_train_data = DataFrame(fitted_model.resid)
residuals_train_data.plot()
pyplot.show()
```

Finally we will use the predict() function to predict what the current. Thus we bring in our current data, which supposedly contains the the data from the DDOS attack.

Our training data: Baseline network Data for a period of a month

Our Testing Data: DDOS attack data

```
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
ddos_predictions = list()
history = new_count_df
for ddos in range(len(ddos_data)):
    model = ARIMA(history, order=(10,1,0))
      fitted_model = model.fit(disp=0)
      output =fitted_model.forecast()
```

We can plot the error that is difference between the forecasted DDOS free network data and the data with the network by computing the mean square error between them

```
pred = output[0]
ddos_predictions.append(pred)
error = mean_squared_error(ddos_data,ddos_predictions)
```

The output of the above is the follwoing plot where the dense line is the forecasted data and the dotted line is the data from the DDOS attack.

Ensemble learning methods

Ensemble learning methods are used to improve performance by taking the cumulative results from multiple models to make a prediction. Ensemble model overcomes the problem of overfitting by considering outputs of multiple models. This helps in overlooking modeling errors from any one model.

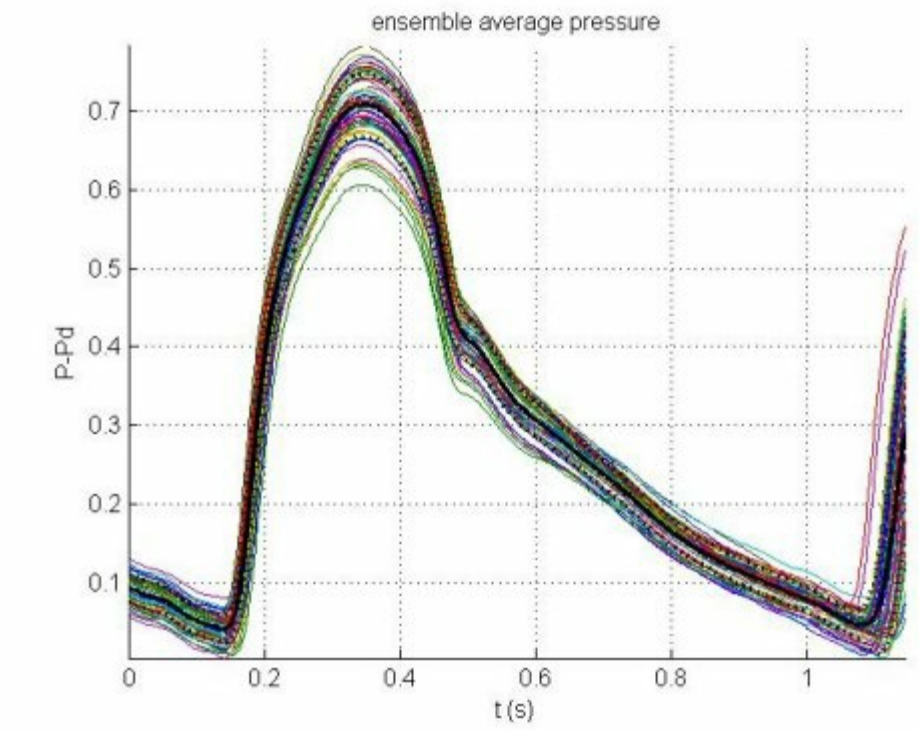
Ensemble learning can however be a problem for time series models because every data point has a time dependency. However if we choose to look at the data as a whole , we can overlook the time dependency components. The time dependency components we can use conventional ensemble methods like bagging, boosting, random forests and so on.

Types of Ensembling

Ensembling of model to derive the best model performance can happen in many ways.

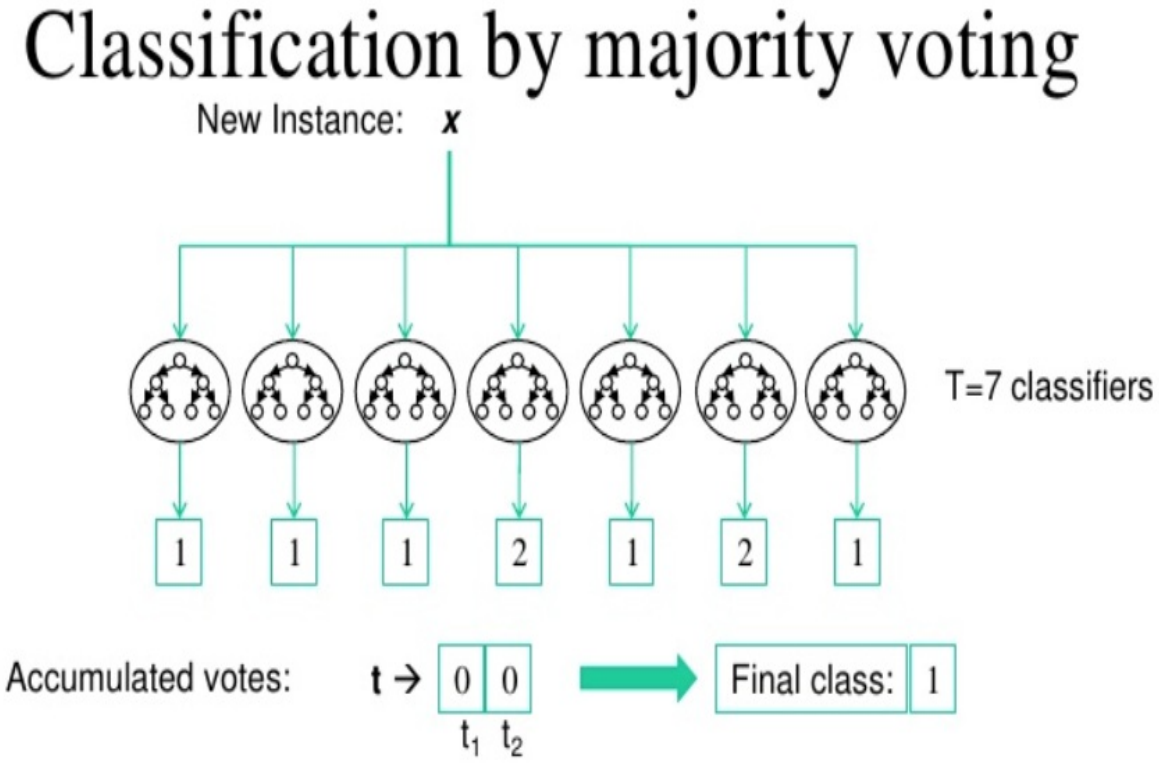
Averaging

In this ensemble method the mean of predictions result is considered from multiple number of predictions that have been made. Here the mean of the ensemble is dependent on the choice of ensemble, hence their value changes from one model to another.



Majority Vote

In this ensemble method, the forecast that gets unanimously voted by multiple models wins and is considered as the end result. For example while classifying an email as spam if alteast three out of four classify a document as span, then it is considered as spam.



Weighted Average

In this ensemble methods , weights are assigned to multiple models and while taking average of each of these predictions the weight is also considered. In this method the models that have more weight recieve more preference.

Weighted Voting.

This is similar to unanimous voting, here some members vote more times than others.

Types of ensemble algorithms

Bagging

These are bootstrap aggregators where equal voting rights have been assigned to every model. Variance of these aggregators are maintained by drawing a random subset when making a decision.

Random Forests are extensions of the bagging method. Random forests is a collection of decision trees that help in classification, regression and decisions.

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
get_values = new_count_df.values
A =get_values[:,0:8]
B =get_values[:,8]
seed = 7
number_of_trees = 50
max_num_features = 2
kfold_crossval = model_selection.KFold(n_splits=10, random_state=seed)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, A, B, cv=kfold_crossval)
print(results.mean())
```

Boosting

Boosting is the type of ensemble learning method where each new learning model is trained by taking instances that have been misclassified by the previous learning algorithm. This process is composed of a series of weak learners but they can classify the entire training data set when they work together. These models often suffer from over fitting issues.

Stacking

If machine learning models were stacked one over the other and each learning model passes its prediction to the model on top such that the model on the top uses the predictions from the previous model layer as input.

Bayesian Parameter Averaging

This is a type of ensemble learning where the Bayesian Parameter Average model approximates the optimal classifier by taking hypothesis from hypothesis space and then applying Bayes' algorithm of them. Here the hypothesis spaces are sampled by using algorithms like the Monte Carlo Sampling, Gibbs Sampling and so on. These are also known as Bayesian Model Averaging.

Bayesiean Model Combination

Bayesian Model Combination(BMC) is modification to the Bayesian Model Averaging (BMA) model. BMC ensembles from space of possible models. BMC provide us with great results. The performance is far better than the Bagging and BMA algorithm. BMC uses cross validation methods to approximate results from the model thus helping to select better ensembles from a collection of ensembles.

Bucket of Models

This approach uses a model selection algorithm to find the best performing model for each use-case. Bucket of models need to be tested across many use cases to derive the best model by weighting and averaging. Similar to the BMC method they choose models in the bucket by methods of cross validation. If the number of usecases are very high the model that takes more time to train should not be taken from the selection. This selection of considering fast learning models is also know as landmark learning.

Cyber security with ensemble techniques

Like other machine learning techniques Ensemble techniques are useful in cyber security. We will go through the same DDOS use case but instead of using a time series model to forecast the attack, we will be using an ensemble method instead.

Voting ensemble method to detect cyber attack.

We start by importing the respective libraries:

```
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

We detect cyber attack via a voting mechanism where we use algorithms like SCV, Decision Tree and Logistic Regression. We finally use the voting classifier to choose the best of the three. Next we create the sub models and pass this through the DDOS data set as follows:

```
voters = []
log_reg = LogisticRegression() # the logistic regression model
voters.append(('logistic', model1))
desc_tree = DecisionTreeClassifier() # the decision tree classifier model
voters.append(('cart', model2))
cup_vec_mac = SVC() # the support vector machine model
voters.append(('svm', model3))
```

For the final voting the voting classifier is invoked as follows:

```
# create the ensemble model
ensemble = VotingClassifier(voters)
```

The final model is chosen by performing a k- fold cross validation

```
results = model_selection.cross_val_score(ensemble, X, Y, cv=kfold)
print(results.mean())
```

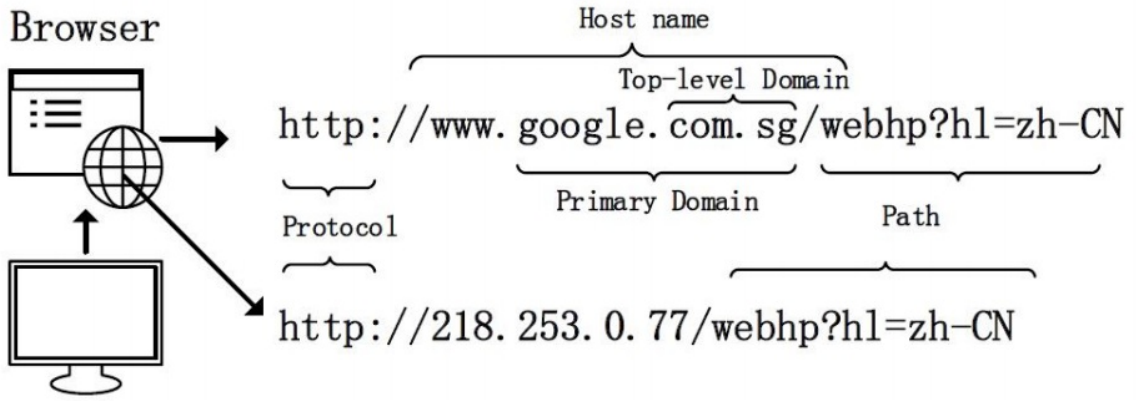
Summary

In this chapter we deal with time series analysis. We will learn different models in the chapter such as stock market predictions, weather forecasting and reconnaissance detection

Segregating legitimate and lousy URLs

A URL stands for a uniform resource locator. A URL is essentially the address of a web page located in the world wide web. URLs are usually displayed in the web browser's address bar. A uniform resource locater confirms to the following address scheme:

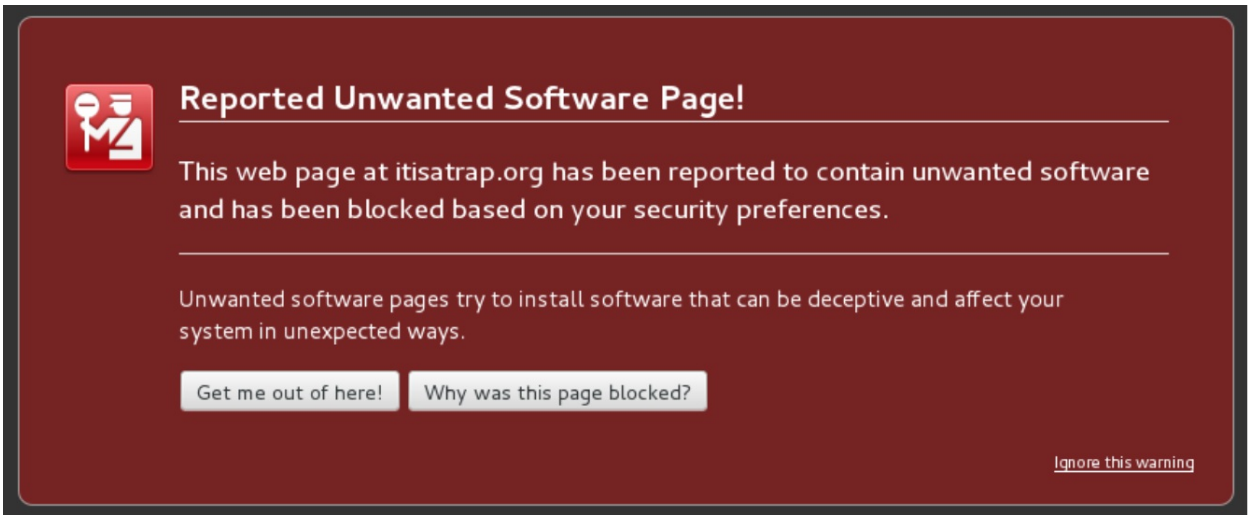
```
scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
```



Url could include either the hyper text transfer protocol (http) or the hyper text transfer secure protocol (https). Other type of protocols include the file transfer protocol(ftp), simple mail transfer protocol or (SMTP) and others like telnet, DNS and so on. A url also consists of the top level domain, hostname, paths and port informations.

What are lousy URLs?

Lousy URLs are URLs that have been created with a malicious intent. They are often the predecessors of cyber attacks that may happen in the near future. Lousy urls hot pretty close to home leaving each one of us very vulnerable to bas sites that we may visit on or without purpose.



Malicious urls lead us to bad websites that either try to sell us counterfeited products like medication, viagra, unsolicited products like watches from Rolex and so on. These websites sell a variety of items like screen savers for your computer and funny pictures.

Bad Urls may also be phishing site that is a site that imitates real website like banks, credit card company websites, but with a sole intent of stealing credentials

The images below shows legitimate Bank of America login page v/s a fake Bank of America login page. What separates the both is that the fake page has a illegitimate URL.

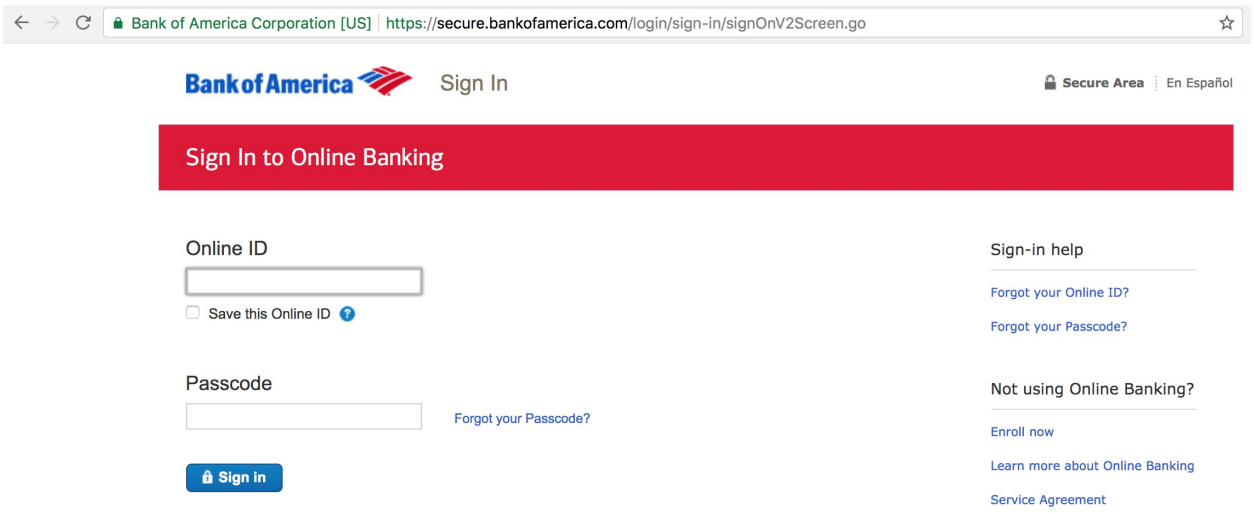


Fig : A real Bank of America login page

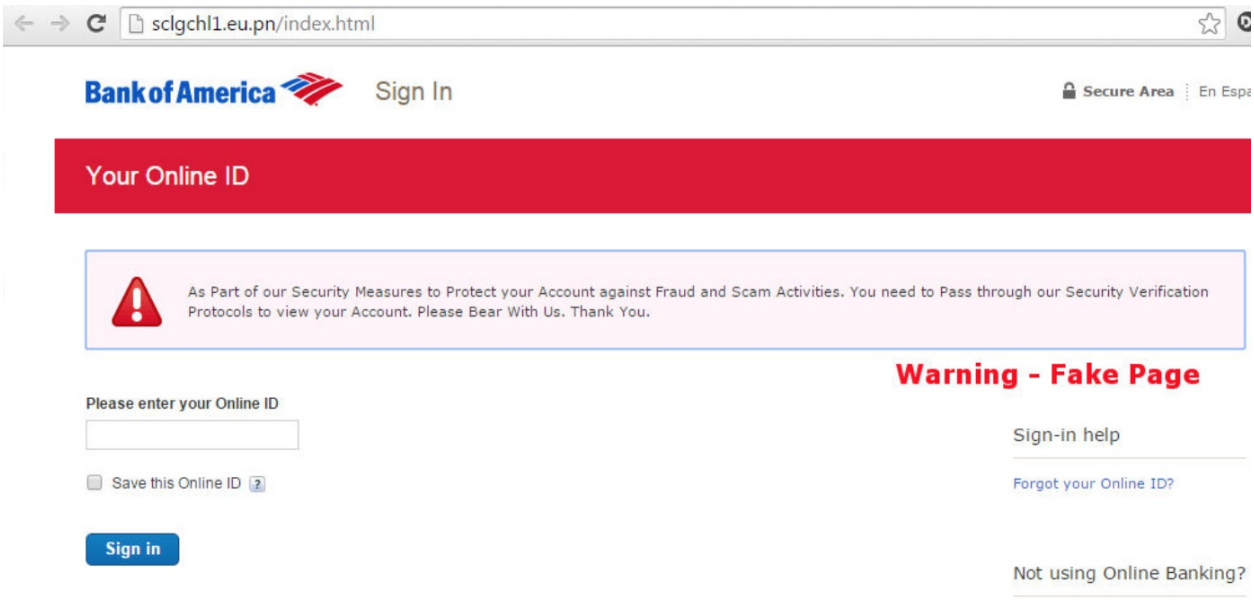
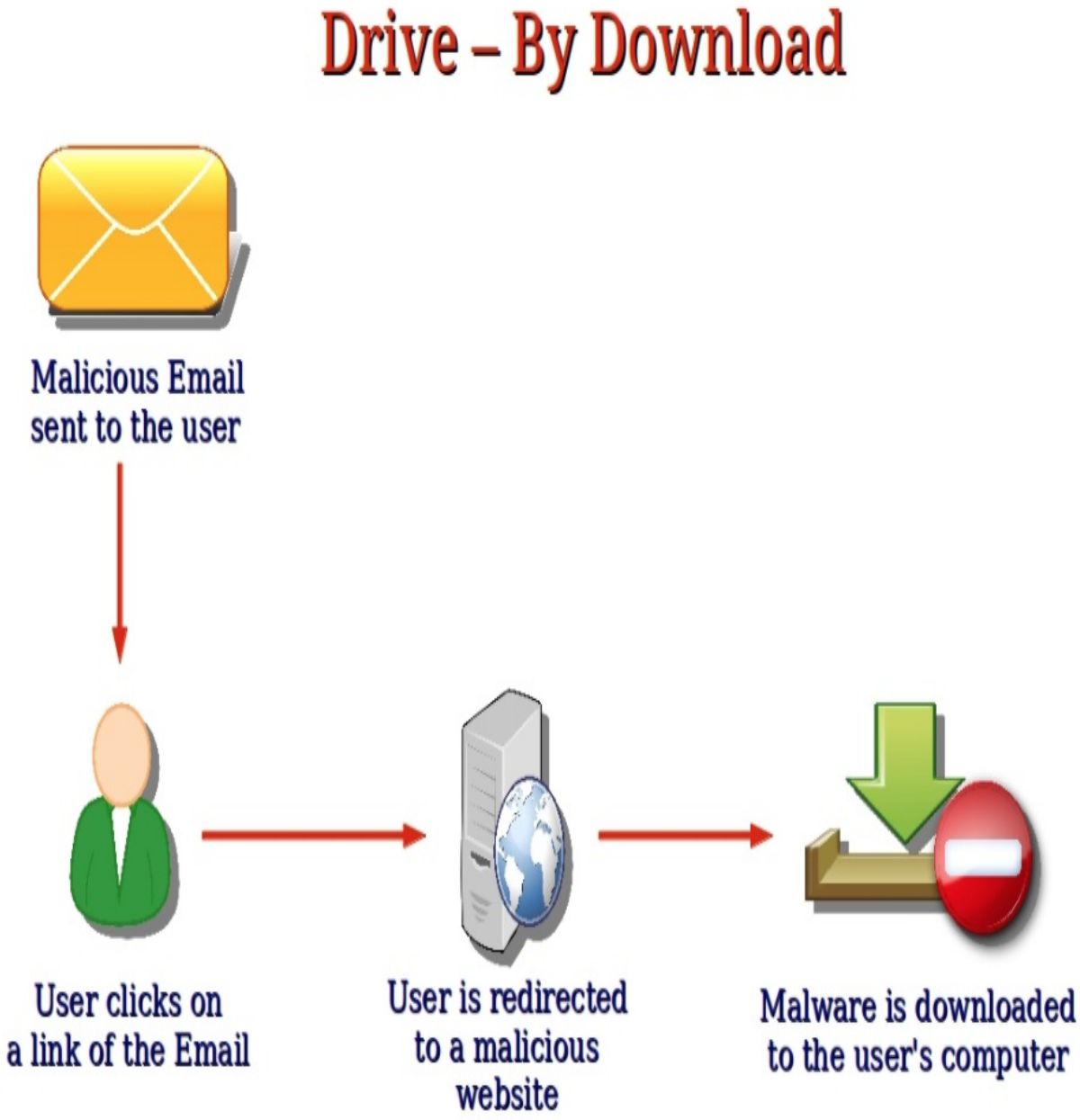


Fig : A fake Bank of America login page

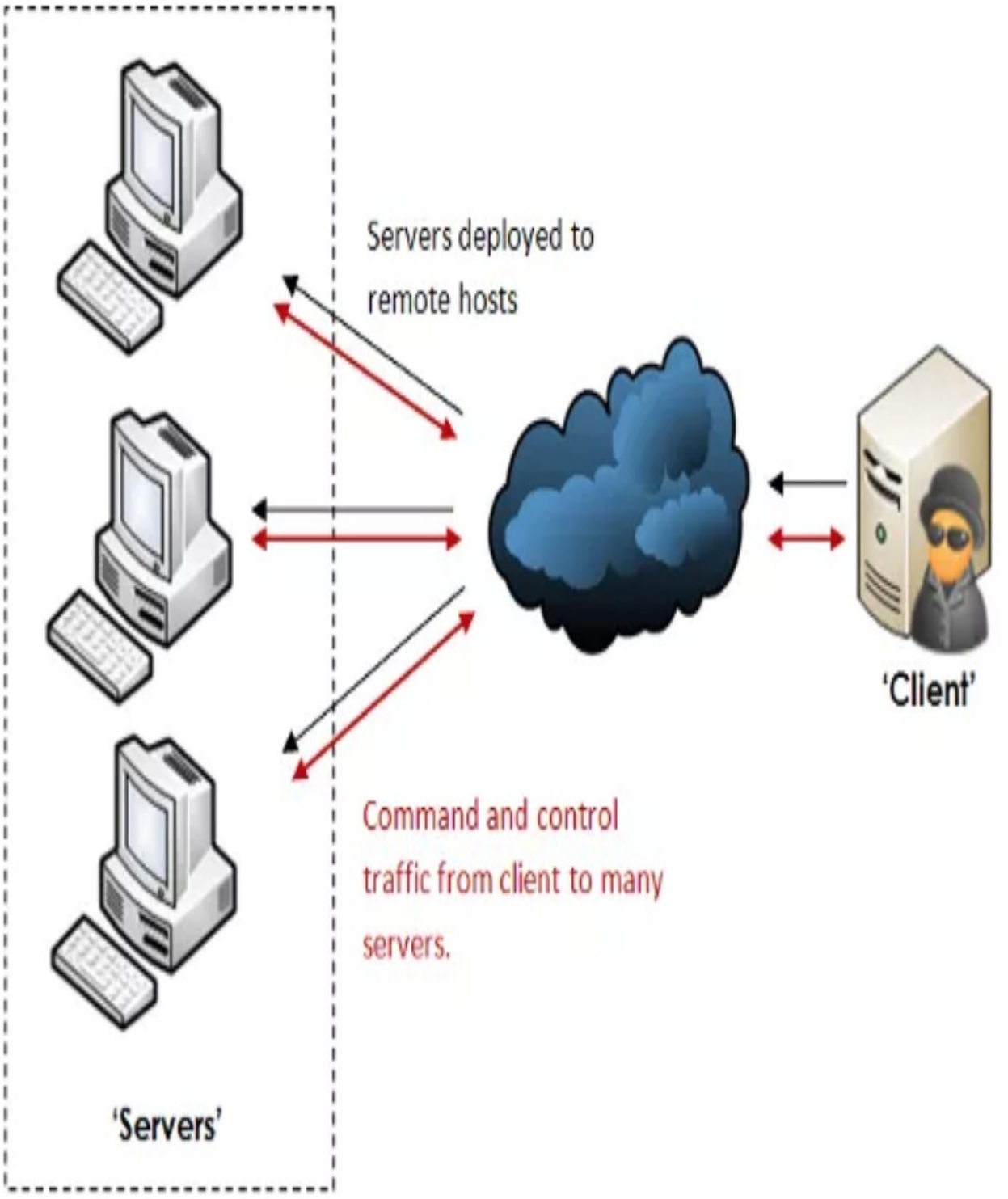
URL blacklisting

These are traditional methods of detecting malicious URLs. Blacklists are static lists of containing an esxhaustive list of URLs that have been identified to be harmful. These urls are usually created by web security companies and agencies. they can be categorized into multiple primary types.

Drive by download urls: These are urls that promote unintended download of software from websites. They could either be downloaded after a naive user has initially clicked on a url, without knowing the consequences that fellow. Drive by downloads could also results from downloads that are carries by malwares that have infected a system, Drive by downloads are by far the most prevalent time of attacks.



Command and Contro Urls: These are urls that have been associated with malwares that make connection to command and control servers. These are different from URLs that can be categorized as malicious as they are always not always a virus that connect to command and control urls to external or remote servers. the connections being inside out.


Servers deployed to
remote hosts
'Client'
Command and control
traffic from client to many
servers.
'Servers'

Phishing URls

These are attacks that steal sensitive information like PAI data by luring or disguising as legitimate or trustworthy urls. Phishing is usually carried through emails or instant messages and direct users to fake website by disguising the urls as legitimate ones.

Using machine learning to detect malicious pages

We have already discussed the different kinds of URLs like benign urls, spam urls and malicious urls. In the exercise below we will categorize urls and there by make a prediction of the type of pages that they would redirect us to. Benign urls always drive us to benign sites. Spam Urls are either lead us to a command and control server or spam website that tries to sell us unsolicited items. Malicious urls are urls that lead us sites that install some kind of a malware to our systems.Since the system does not actually visit the pages of the websites that the url points to, we are able to save a lot in terms of latency and get a better performance.

Data for the analysis

We gather data from different sources and are able to create a data set with approximately 1000 urls. These urls are pre labelled in the respective classes benign, spam and malicious. Below is a snippet from our URL data set.

```
http://uol.com.br,0
http://google.pl,0
http://ebay.co.uk,0
http://netflix.com,0
http://dailymotion.com,0
http://cnet.com,0
http://delta-search.com,0
http://dailymail.co.uk,0
http://rakuten.co.jp,0
http://aliexpress.com,0
http://aol.com,0
http://dce.edu,0
http://google.com,0
http://stackoverflow.com/questions/8551735/how-do-i-run-python-code-from-sublime-text-2,0
http://paypal.com.webscr.cmd.login.submit.dispatch.5885d80a13c0db1f8e263663d3faee8db2b24f7b84f1819343fd6c338b1d9d.222studio.com
http://paypal-manager-login.net/konflikt/66211165125/,1
http://paypal-manager-loesung.net/konflikt/66211165125/,1
http://paypal-manager-account.net/konflikt/66211165125/,1
http://www.paypal-manager-account.net/konflikt/6222649185/index.php,1
http://www.paypal-manager-login.net/konflikt/79235228200/index.php,1
http://paypal.com.laveaki.com.br/PayPal.com/,1
http://paypal.com.client.identifiant.compte.clefs.informations.upgarde.mon.compte.personnel.ghs56hge556rg4h6qe4th654f84e84r8e.h
http://msd2003.com/rche/index.htm,1
http://paypal-manager-loesung.net/konflikt/6624985147/index.php,1
http://eadideal.com.br/conteudos/material/66/Netzro-Login.html,1
http://tahmid.ir/user/?cmd=_home&amp;dispatch=5885d80a13c0db1f8e&amp;ee=669498ee5d0b4b381fd2bb1cceb52112,1
http://208.115.247.198/paypal.com.au.434324.fdsfsd32423423.fadfafas.3423423432fsfdafsdfsd/index1.php,1
http://www.paypal-manager-service.net/konflikt/785549116/index.php?webapps=/mpp/verkaufen,1
http://www.nervemobilization.com/update/06bb5b5bd8b1be54b78648f4993a1de1/protect.html,1
http://www.nervemobilization.com/update/15f58e7e4a736e2cd442b733e8755716/protect.html,1
http://sionaviatur.ru/wp-includes/syystemyj/index50.htm,1
```

Feature extraction

Since the data in hand is structured and pre labelled, we can directly move on to extract the features from the data. We will primarily extract certain lexical features, host based features and popularity based features.

Lexical features

Lexical features are derived by analysis of the lexical unit of sentences. In lexical semantics it is composed of full words or semi formed words. We will analyze the lexical features in the url and extract it in accordance to the urls that are available. We extract the different url components like the address comprising of the hostname, the path and so on.

We start with importing the headers

```
from url parse import urlparse
import re
import urllib2
import urllib
from xml.dom import minidom
import csv
import pygeoip
```

Once done we start with importing the necessary packages we then start with tokenizing the urls. Tokenizing is the process of chopping the url into several pieces. A token refers to a the part that have been broken down into a sequence. The token when taken together are used for semantic processing.

Example of tokenization

The quick brown fox jumps over the lazy dog

Token are :
The
quick
brown
fox
jumps
over
the
lazy

dog

Before we go ahead and start tokenizing urls we need to check if they are IP addresses. def

```
get_IPaddress(tokenized_words):
    count=0;
    for element in tokenized_words:
        if unicode(element).isnumeric():
            count= count + 1
        else:
            if count >=4 :
                return 1
            else:
                count=0;
    if count >=4:
        return 1
    return 0
```

We move on to tokenizing the URLs.

```
def url_tokenize(url):
    tokenized_word=re.split('\W+',url)
    num_element = 0
    sum_of_element=0
    largest=0
    for element in tokenized_word:
        l=len(element)
    sum_of_element+=l
```

For empty element exclusion in average length:

```
        if l>0:
             num_element+=1
             if largest<l:
                 largest=l
 try:
     return [float(sum_of_element)/num_element,num_element,largest]
 except:
     return [0,num_element,largest]
```

Malicious sites that use phishing urls to lure people are usually longer in length.Each token separated by dot. We look for several predefined knowledge about malicious emails. We search for these patterns in the tokens

1. We look for .exe files in the token of the data. If the token showed that the url contained exe files pointers in the url, we flag it.

```
def url_has_exe(url):
 if url.find('.exe')!=-1:
     return 1
 else :
     return 0
```

2. We look for common words that are associated with phishing. We count the presense of these words like

'confirm', 'account', 'banking', 'secure', 'viagra', 'rolex', 'login', 'signin'

```
def get_sec_sensitive_words(tokenized_words):
    sec_sen_words=['confirm', 'account', 'banking', 'secure', 'viagra', 'rolex', 'login', 'signin']
    count=0
    for element in sec_sen_words:
        if(element in tokenized_words):
            count= count + 1;
    return count
```

Web Content Based Features

We look for features that usually available in malicious pages like :

1. Count of html tags in the web page
2. Count of hyperlinks in the web page
3. Count of iframes in the web page

```
def web_content_features(url):
    webfeatures={}
    total_count=0
    try:
        source_code = str(opener.open(url))
        webfeatures['src_html_cnt']=source_code.count('<html')
        webfeatures['src_hlink_cnt']=source_code.count('<a href=')
        webfeatures['src_iframe_cnt']=source_code.count('<iframe')
```

We also count the number of suspicious java script objects.

1. Count of evals
2. Count of escapes
3. Count of links
4. Count of underescapes
5. Count of exec() functions
6. Count of search functions

```
        webfeatures['src_eval_cnt']=source_code.count('eval(')
        webfeatures['src_escape_cnt']=source_code.count('escape(')
        webfeatures['src_link_cnt']=source_code.count('link(')
        webfeatures['src_underescape_cnt']=source_code.count('underescape(')
        webfeatures['src_exec_cnt']=source_code.count('exec(')
        webfeatures['src_search_cnt']=source_code.count('search(')
```

We also count the number of times html, hlink and iframe in the web feature keys.

```
 for key in webfeatures:
    if(key!='src_html_cnt' and key!='src_hlink_cnt' and key!='src_iframe_cnt'):
        total_count=total_count + webfeatures[key]
    webfeatures['src_total_jfun_cnt']=total_count
```

We also look for other web features and handle the exceptions:

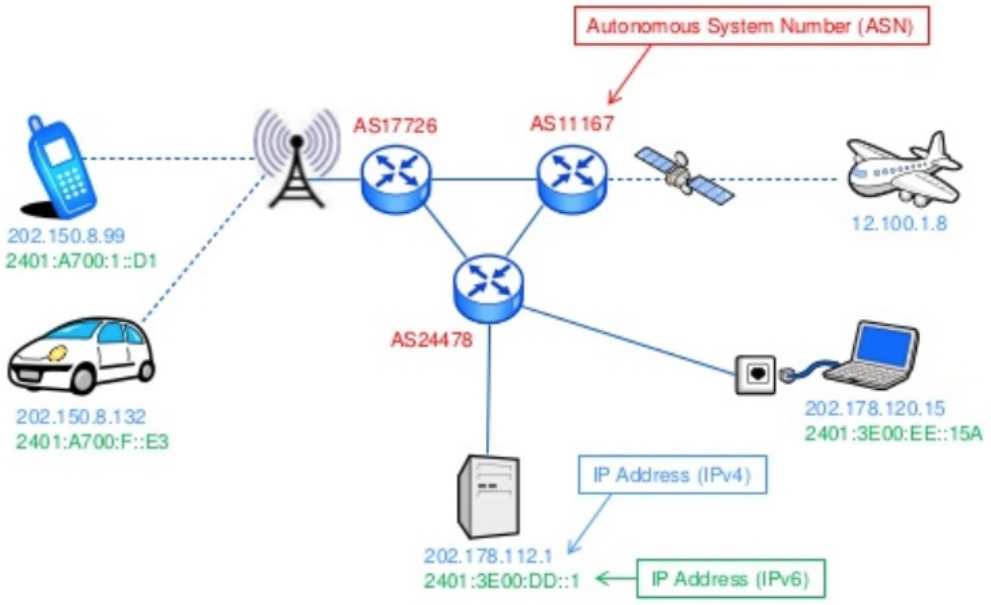
```
    except Exception, e:
        print "Error"+str(e)+" in downloading page "+url
        default_value=nf

        webfeatures['src_html_cnt']=default_value
        webfeatures['src_hlink_cnt']=default_value
        webfeatures['src_iframe_cnt']=default_value
        webfeatures['src_eval_cnt']=default_value
        webfeatures['src_escape_cnt']=default_value
        webfeatures['src_link_cnt']=default_value
        webfeatures['src_underescape_cnt']=default_value
        webfeatures['src_exec_cnt']=default_value
        webfeatures['src_search_cnt']=default_value
        webfeatures['src_total_jfun_cnt']=default_value

    return webfeatures
```

Host based features

Hosting services are internet services that allow users to make a website available to the World Wide Web. Hosting service providers away server places. A lot of the times a single server is rented out to multiple websites. We will be find out the ip addresses of each of urls that we inspect and we will also look up the Autonomous System Number. AS numbers are a collection of IP routing prefixes that are controlled by central network operators. We use the ASN information to look for sites that have already been fall under Bad ASN category.




```
def getASN(host_info):
    try:
        g = pygeoip.GeoIP('GeoIPASNum.dat')
        asn=int(g.org_by_name(host_info).split()[0][2:])
        return asn
    except:
         return nf
```

Site popularity features

We use Alexa ranking for website as one of our features. Alexa is ranking of websites based on the popularity by the unique number of individuals who visit the site. We use the Alexa popularity rank for each website. The basic idea was that highly popular sites are usually non malicious.

The top 10 items of the Alexa looks like :

Google	google.com	1
YouTube	youtube.com	2
Facebook	facebook.com	3
Baidu	baidu.com	4
Wikipedia	wikipedia.org	5
Reddit	reddit.com	6
Yahoo!	yahoo.com	7
Google India	google.co.in	8
Tencent QQ	qq.com	9
Amazon	amazon.com	10

The following python function is used to detect the popularity.

```
def site_popularity_index(host_name):
    xmlpath='http://data.alexa.com/data?cli=10&dat=snbamz&url='+host_name
    try:
        get_xml= urllib2.urlopen(xmlpath) # get the xml
        get_dom =minidom.parse(get_xml) # get the dom element
        get_rank_host=find_ele_with_attribute(get_dom,'REACH','RANK')
        ranked_country=find_ele_with_attribute(get_dom,'COUNTRY','RANK')
        return [get_rank_host,ranked_country]
    except:
        return [nf,nf]
```

Summary

In this chapter we learn about what are URLs and what do you mean by lousy URLS. We also learned methods to identify malicious URLs, blacklisted URLs and Phishing URls. Later we used machine learning to detect malicious pages using different features.

Knocking Down Captchas

Coming soon...

Using Data Science to Catch Email Frauds and Spams

Coming soon...

Efficient Network Anomaly Detection Using K Means

Coming soon...

Decision Tree and Context Based Malicious Event Detection

Coming soon...

Catching Impersonators and Hackers Red Handed

Coming soon...

Speeding Things Up with GPU

Coming soon...

Change the Game with TensorFlow

Coming soon...

Financial Frauds and How Deep Learning Can Mitigate them

Coming soon...